END-TO-END AVAILABLE BANDWIDTH ESTIMATION AND ITS APPLICATIONS

A Thesis
Presented to
The Academic Faculty

by

Manish Jain

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
May 2007

END-TO-END AVAILABLE BANDWIDTH ESTIMATION AND ITS APPLICATIONS

Approved by:

Constantine Dovrolis, Advisor
College of Computing
Georgia Institute of Technology

Mostafa Ammar
College of Computing
Georgia Institute of Technology

Karsten Schwan
College of Computing
Georgia Institute of Technology

Peter Steenkiste
Department of Computer Science
Carnegie Mellon University

Ellen Zegura
College of Computing
Georgia Institute of Technology

Date Approved: November 27th, 2006

To my family

ACKNOWLEDGEMENTS

A line I heard from my favorite TV series - *Life is a series of rooms and who we get stuck in those rooms with, adds up to what our lives are.* The people I *was stuck in rooms with* in the last five years made this time in my life truly amazing and memorable. I am taking this opportunity to thank all those people.

The first and most important person is my advisor and mentor Prof. Constantine Dovrolis. His role in this journey has been identical to that of a parent who hold their kid's finger and are always there to break the fall while the kid is learning to walk. This journey became possible largely because of all the support, encouragement and belief that he has shown in me during this time. His enthusiasm has always inspired me and I have learnt a great deal from him during these years. I feel really fortunate to have him as my mentor. I am also thankful to him for continuously supporting me as Research Assistant through his grants.

I would also like to thank the members of my committee Mostafa Ammar, Karsten Schwan, Peter Steenkiste and Ellen Zegura. Prof. Ammar, Prof. Schwan and Prof. Zegura have always taken interest in my thesis and been very encouraging about my work. Prof. Steenkiste provided valuable feedback in improving my thesis as the external member of my committee.

During my internship at HP labs, I had the opportunity to work with Sujata Banerjee and Puneet Sharma. It was a great learning experience and I am thankful for the opportunity.

I have had the good fortune of interacting and working with some really wonderful people in the NTG group and I thank them all. Abhishek Kumar, Amogh Dhamdhere, Hyewon Jun, Qi He, Pradnya Karbhari, Ruomei Gao, Sridhar Srinivasan and Srinivasan Seetharaman have all made my years at Georgia Tech very enjoyable.

Two people, Ravi Prasad and Yarong Tang, deserve special mention and I am very

thankful for their friendship. Ravi, who has been a friend of mine for several years, is an amazing person to be around and work with. All the time that we spent on freeciv sessions, squash, and long discussions about life in general (and work) have been fun and memorable. I have always been able to count on him in the time of need and I am truly grateful for that. Yarong has been a wonderful and dependable friend during my my years at Gatech.

Finally, I would like to thank my parents and younger brother for their love and support at every step of my life. This thesis is dedicated to them.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES x

LIST OF FIGURES xi

SUMMARY xiv

I INTRODUCTION 1

1.1 Early work on Available Bandwidth Estimation 3

1.2 Contributions 4

1.2.1 Avail-bw Estimation Methodology: *SLoPS* 5

1.2.2 Average Available Bandwidth: Algorithm and Tool 5

1.2.3 Available Bandwidth Variation Range: Algorithm and Tool 5

1.2.4 Effect of Interrupt Coalescence on Bandwidth Estimation Tools . 6

1.2.5 Misconceptions in Available Bandwidth 6

1.2.6 Video Streaming using Avail-bw 6

1.3 Thesis Organization 7

II SELF LOADING PERIODIC STREAM (SLOPS) 8

2.1 Definitions 8

2.2 Self-Loading Periodic Streams 9

2.2.1 SLoPS with fluid cross traffic 9

2.2.2 An iterative algorithm to measure A 13

2.2.3 SLoPS with real cross traffic 13

2.3 Summary 16

III AVERAGE AVAIL-BW ESTIMATION: PATHLOAD 17

3.1 Measurement tool: Pathload 18

3.1.1 Clock issues 18

3.1.2 Stream parameters 18

3.1.3 Detecting an increasing OWD trend 19

3.1.4 Fleets of streams 20

3.1.5 Rate adjustment algorithm . . . 22
3.1.6 Measurement latency . . . 23
3.1.7 Response to packet losses . . . 23
3.1.8 Detection of a sender context switch . . . 24
3.1.9 Detection of a receiver context switch . . . 25
3.2 Verification . . . 26
3.2.1 Simulation results . . . 26
3.2.2 Experimental results . . . 31
3.3 Available bandwidth dynamics . . . 35
3.3.1 Variability and load conditions . . . 36
3.3.2 Variability and statistical multiplexing . . . 37
3.3.3 The effect of the stream length . . . 39
3.3.4 The effect of the fleet length . . . 41
3.4 TCP and available bandwidth . . . 41
3.5 Is Pathload intrusive? . . . 45
3.6 Summary . . . 47
IV AVAILABLE BANDWIDTH VARIATION RANGE ESTIMATION: PATHVAR 49
4.1 Variation Range of Avail-bw . . . 49
4.1.1 Definitions . . . 50
4.1.2 Main Contributions and Overview . . . 52
4.2 Percentile sampling . . . 53
4.2.1 Basic idea . . . 54
4.2.2 How large should N be? . . . 55
4.3 Non-parametric estimation . . . 57
4.3.1 Algorithm . . . 57
4.3.2 Estimation with non-stationary load in single-hop path . . . 59
4.4 Parametric estimation . . . 62
4.4.1 Algorithm . . . 62
4.4.2 Validation examples . . . 65
4.5 Pathvar . . . 66
4.5.1 Testbed examples . . . 68

4.5.2 Internet Experiments . . . 73
4.6 Variability factors . . . 74
4.6.1 Effect of tight link utilization . . . 76
4.6.2 Statistical multiplexing effects and scaling models . . . 77
4.6.3 Effect of measurement timescale . . . 80
4.7 Pathvar Overhead and Latency . . . 82
4.8 Summary . . . 84
V EFFECTS OF INTERRUPT COALESCENCE ON NETWORK MEASUREMENTS 85
5.1 Description of Interrupt Coalescence . . . 86
5.2 Effects of Interrupt Coalescence . . . 88
5.2.1 Capacity Estimation. . . . 88
5.2.2 Available Bandwidth Estimation. . . . 91
5.2.3 Passive measurements . . . 93
5.2.4 TCP self-clocking . . . 93
5.3 Summary . . . 94
VI APPLICATION OF AVAIL-BW ESTIMATION: VIDEO STREAMING . . . 95
6.1 Introduction . . . 95
6.2 VDN architecture and in-band measurements . . . 97
6.3 Path selection schemes . . . 99
6.4 Experimental setup . . . 100
6.5 Results . . . 103
6.6 Path switching versus FEC . . . 107
6.7 Summary . . . 108
VII RECENT WORK IN AVAIL-BW ESTIMATION . . . 109
7.1 Avail-bw estimation at a single link with fluid traffic . . . 109
7.1.1 Direct probing . . . 111
7.1.2 Iterative probing . . . 111
7.2 Classification . . . 111
7.3 Fallacies and pitfalls [1] . . . 114
7.3.1 Pitfall: Ignoring the variability of the avail-bw process. . . . 114

7.3.2 Pitfall: Ignoring the relation between probing stream duration and averaging time scale. 115
7.3.3 Fallacy: Faster estimation is always better. 116
7.3.4 Fallacy: Packet pairs are as good as packet trains. 116
7.3.5 Pitfall: Estimating the tight link capacity with end-to-end capacity estimation tools. 117
7.3.6 Fallacy: Increasing One-Way Delays is equivalent to $R_o < R_i$. . . 118
7.3.7 Fallacy: Iterative probing converges to a single avail-bw estimate (as opposed to a variation range). 119
7.3.8 Pitfall: Ignoring the effects of cross traffic burstiness. 120
7.3.9 Pitfall: Ignoring the effects of multiple bottlenecks. 121
7.3.10 Pitfall: Evaluating the accuracy of avail-bw estimation through comparisons with bulk TCP throughput. 122
7.4 Summary 123
VIII CONTRIBUTIONS AND FUTURE WORK 124
8.1 Research Contributions 124
8.2 Future Directions 125
REFERENCES 127
VITA 134

LIST OF TABLES

1 Measurement hosts and their locations. 32

2 Effect of cross traffic packet size L_c on the relative error β for four sample sizes k. 117

LIST OF FIGURES

1 The e2e avail-bw in this 3-hop path is 6 Mbps. 2

2 OWD variations for a periodic stream when $R > A$. 14

3 OWD variations for a periodic stream when $R < A$. 14

4 OWD variations for a periodic stream when $R \bowtie A$. 15

5 Comparison of PCT and PDT for a stream with increasing OWD trend. . . 20

6 PCTs for three fleets (20 streams each). 22

7 PDTs for three fleets (20 streams each). 23

8 A context switch at the sender. 24

9 A context switch at the receiver. 25

10 Simulation topology. 26

11 Simulation results for different values of fraction f. 27

12 Simulation results for different values of PDT threshold. 28

13 Simulation results for different traffic types and tight link loads. 29

14 Simulation results for different non-tight link loads. 29

15 Simulation results for different path tightness factors β. 30

16 Example of an MRTG graph for a 14Mbps link. The x-axis is measured in hours. 32

17 Verification experiment in Path #1 33

18 Verification experiment in Path #2 34

19 Verification experiment in Path #3 35

20 Verification experiment in Path #4 36

21 Verification experiment in Path #5 37

22 Pathload evaluation result from Shriram et al. [80] 38

23 Variability of avail-bw in different load conditions. 38

24 Variability of avail-bw in different paths. 39

25 The effect of K on the variability of the avail-bw. 40

26 The effect of N on the variability of the avail-bw. 40

27 Available bandwidth and BTC throughput. 42

28 RTT measurements during the experiment of Figure 27. 43

29 Available bandwidth measurements. 45
30 RTT measurements during the experiment of Figure 29. 46
31 Measurement latency in a 100 msec round-trip time path. 47
32 Average sending rate in a 100 msec round-trip time path. 48
33 Top: Time series of the avail-bw process at an OC-3 link in two measurement time scales. Bottom: Empirical CDFs of the two time series. 50
34 N_{min} as a function of ρ for $p = 0.9$ and $\epsilon = 0.05$. 56
35 Actual and estimated variation range when b=.05 60
36 Actual and estimated variation range when b=.15 61
37 Estimated and actual 10-90% variation range with Gaussian traffic. 65
38 Estimated and actual 10-90% variation range with non-Gaussian traffic. . . 66
39 Pathvar experiment with non-parametric algorithm. 68
40 Pathvar experiment with parametric algorithm. 69
41 Non-parametric and parametric estimates of 20% percentile for τ=40msec. . 70
42 Non-parametric and parametric estimates of 20% percentile for τ=140msec. 70
43 Non-parametric and parametric estimates of 20% percentile for low vertical aggregation. 71
44 Non-parametric and parametric estimates of 20% percentile for high vertical aggregation. 72
45 Variation range estimates at the Internet path from Georgia Tech to University of Ioannina. 73
46 Variation range estimates at the Internet path from University of Crete to Georgia Tech. 74
47 Effect of tight link utilization on the variation range width (left Y-axis) and CoV (right Y-axis). 76
48 Effect of capacity scaling on the variation range width (left Y-axis) and CoV (right Y-axis). 78
49 Effect of flow scaling on the variation range width (left Y-axis) and CoV (right Y-axis). 79
50 Effect of time scale τ on the variation range width. 81
51 Measurement latency for the non-parametric algorithm. 82
52 Average sending rate for the non-parametric algorithm. 83
53 Average sending rate for the parametric algorithm. 83
54 Illustration of IC with *RxAbsIntDelay* (Case I) and with *RxIntDelay* (Case II). 87

55 Cumulative dispersion in an 100-packet train with and without IC. 89
56 The signatures of context switching and interrupt coalescence in the cumulative dispersion of a packet train. 90
57 OWDs in 100-packet train with and without IC. 91
58 CDF of ACK interarrivals in a 10-second TCP connection over a GigE path. 94
59 VDN architecture. 98
60 Shaping of video packets to a probing rate R_p for avail-bw estimation. . . . 99
61 Testbed. 100
62 Two NLANR cross traffic traces. 102
63 VQM scores for the four path selection schemes. 102
64 Path switching frequency. 104
65 User-abort probability. 104
66 Cross traffic traces with instances of non-stationarity. 105
67 VQM scores for the PSC trace. 106
68 VQM scores for the AMP trace. 106
69 Path switching versus FEC. 107
70 Relative error β of the sample mean m_A for three averaging time scales. . . 114
71 The probing stream duration controls the averaging time scale τ. 116
72 OWDs for two probing streams of 160 packets. 118
73 Variation range of an avail-bw sample path. 119
74 Effect of cross traffic burstiness. 120
75 Effect of multiple tight links. 121
76 TCP throughput compared to avail-bw. 123
77 Number of Pathload downloads by unique users since April 2003. 125

SUMMARY

The Internet continues to evolve without a mechanism to provide performance guarantees or explicit feedback to applications. In this context, the only way to monitor the state of the network is through end-to-end measurement and inference. The *available bandwidth* (avail-bw) is considered an important metric to characterize the dynamic state of a network path. Even though its end-to-end measurement has been the focus of extensive research during the last 15 years, it is only recently that its direct measurement has become possible, largely as a result of our research.

In this thesis, we focus on developing estimation methodologies and tools for measuring the avail-bw accurately, quickly, and without being intrusive to the rest of the network traffic. First, we present an end-to-end methodology, called Self-Loading Periodic Streams (SLoPS), to determine whether the avail-bw is greater than a given rate based on the one-way delays of periodic packet streams. The basic idea in SLoPS is that the one-way delays of a periodic packet stream show an increasing trend when the stream's rate is higher than the avail-bw. The SLoPS methodology is the cornerstone of the two avail-bw estimation tools, pathload and pathvar, that we develop to estimate the average avail-bw and its variation range, respectively. *Pathload* uses SLoPS coupled with a binary probing algorithm to estimate the average avail-bw. *Pathvar* implements two techniques, non-parametric and parametric, to estimate the variation range of avail-bw in a user specified time scale.

We verify the accuracy of the tools using simulation, testbed and Internet experiments. We identify that Interrupt Coalescence (IC), a technique implemented in network cards to reduce interrupt overhead, is a factor affecting the accuracy of most avail-bw estimation tools. We present an algorithm which can detect and discard the packets that were affected by IC, thereby minimizing the effect of IC on avail-bw measurements. The key idea in this algorithm is based on the signature that IC leaves on the dispersion and one-way delays of packet streams.

We also use Pathvar to evaluate the variability of the avail-bw with both simulations and measurements in Internet paths. We identify four factors that play a crucial role in the variability of the avail-bw: traffic load, number of competing flows, rate of competing flows, and the measurement time scale.

We then review several avail-bw estimation techniques and tools developed in the last 4-5 years. We argue that some key issues regarding the avail-bw definition, estimation, and validation remain vague or misinterpreted. We identify ten misconceptions, in the form of fallacies or pitfalls, that we consider as most important and explain them, with analysis and/or simulations, to enhance the understanding of avail-bw as a useful metric.

Finally, we present an application of avail-bw estimation in video streaming. Specifically, we show that the avail-bw measurements can be used in the dynamic selection of the best possible overlay path. We compare the performance of various measurement-based path selection techniques for video streaming in terms of perceived video quality. The avail-bw based scheme results in better perceived video quality than path selection algorithms that rely on jitter or loss-rate measurements.

CHAPTER I

INTRODUCTION

Since the inception of ARPAnet in the 1960s, networks running TCP/IP technology grew from a few networks to approximately 20000 networks today and from a few hundred hosts to over 400 million hosts [29]. During the same time, the number and types of services/applications that are available over the Internet have grown from news, gopher and email to interactive video conferencing, video streaming, and P2P file transfer. All these applications have different Quality of Service (QoS) requirements from the network. For example, video streaming requires low jitter and loss rate, while P2P file transfers care more about throughput. Similarly, different classes of Internet users have different requirements. Companies that use VoIP in place of conventional telephone have more stringent QoS requirements from the network than a user at home using the Internet mostly for web browsing.

A *network-centric* approach to support the different QoS requirements of users/applications is to build *service differentiation* mechanisms in the Internet infrastructure. The Intserv [13, 89, 91] and Diffserv [62, 10] architectures were developed within IETF during the 1990s to provide such service differentiation. The main premise in these approaches was that different applications have different resource requirements and the objective of the network should be to provide different QoS guarantees or assurances. Intserv was an end-to-end (e2e) approach that provides service differentiation by reserving resources on an e2e basis for each flow. Diffserv, on the other hand, aggregates flows into few classes at the network edges and provides service differentiation between different classes of service on a per-hop basis. However, deployment of both these approaches has been very limited so far. This is due to several factors including signaling overhead and inter-domain deployment issues with Intserv, and the inability to provide e2e assurances with Diffserv.

Another approach to deal with the diverse QoS requirements of users and applications

is to assume that the network only provides a common basic service for all traffic and build mechanisms in the end-hosts or applications to dynamically detect and adapt to changing network conditions. Such an *end-centric* approach requires that the hosts or applications have a way to infer network conditions. In the past, there have been several proposals that required routers to send information to sources/senders about the network state [74, 43]. These proposals have, however, not been deployed and/or enabled in the routers, and the Internet still remains a *black box* from the end-host perspective. The lack of explicit feedback from the network infrastructure has been the main motivation for researchers to focus on designing algorithms and tools that can monitor the network state through e2e measurement and inference.

End-to-end network path performance can be characterized by several network-level metrics: loss rate, delay, TCP bulk transfer capacity (BTC), capacity and *avail-bw*. All the metrics, with the exception of avail-bw, have one limitation: they are not a direct indicator of traffic load; losses can be random and/or happen after a link has been congested; delay may be dominated by propagation latencies; BTC depends on several factors such as send/receive windows size and TCP stack; the capacity, by definition, does not take traffic load into account. The avail-bw, on the hand, is a direct measure of the additional load that a network path can carry before it becomes saturated. Figure 1 illustrates the basic concept of the e2e avail-bw in a three-hop path.

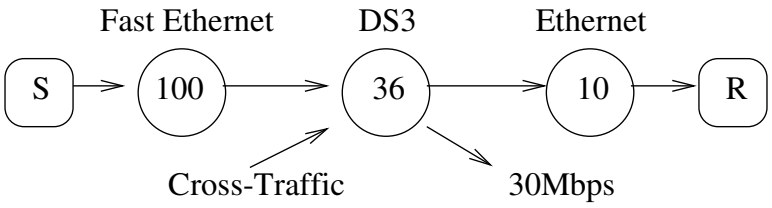


Figure 1: The e2e avail-bw in this 3-hop path is 6 Mbps.

The concept of *avail-bw* is not something new and has been of central importance throughout the history of packet networks, in both research and practice. In the context of transport protocols, the robust and efficient use of avail-bw has always been a major

issue, including Jacobson's TCP [30]. The avail-bw is also a crucial parameter in capacity provisioning, routing and traffic engineering, QoS management, streaming applications, server selection, and in several other areas.

Researchers have been trying to create e2e measurement algorithms for avail-bw over the last 15 years. From Keshav's *packet pair* [45] to Carter and Crovella's *cprobe* [14], the objective was to measure e2e avail-bw *accurately*, *quickly*, and without affecting the traffic in the path, i.e., *non-intrusively*. What made the measurement of avail-bw hard is, first, that there was no consensus on how to precisely define it, second, that it is a dynamic metric, and third, that it exhibits high variability in a wide range of timescales.

1.1 Early work on Available Bandwidth Estimation

Although there are several bandwidth estimation tools, most of them measure capacity rather than avail-bw. Specifically, *pathchar* [31], *clink* [18], *pchar* [57], and the *tailgating* technique of [49] measure *per-hop capacity*. Also, *bprobe* [14], *nettimer* [48], *pathrate* [16], and the *PBM* methodology [70] measure *end-to-end capacity*.

Allman and Paxson noted that an avail-bw estimate can give a more appropriate value for the *ssthresh* variable, improving the slow-start phase of TCP [4]. They recognized, however, the complexity of measuring avail-bw from the timing of TCP packets, and focused instead on capacity estimates.

The first tool that attempted to measure avail-bw was *cprobe* [14]. *cprobe* estimated the avail-bw based on the dispersion of long packet trains at the receiver. A similar approach was taken in *pipechar* [41]. The underlying assumption in these works is that the dispersion of long packet trains is inversely proportional to the avail-bw.

Dovrolis et al. [16], however, showed that the previous assumption is not true. The dispersion of long packet trains does not measure the avail-bw in a path; instead, it measures a different throughput metric that is referred to as *Asymptotic Dispersion Rate* (ADR). [16] also showed a result for a single-hop path that can be used to determine the available bandwidth given the input stream rate, output stream rate and narrow link capacity. Though this result is only valid for single hop path, it has been used by several techniques, described

in chapter 7 as *direct probing* approach, that have been proposed in the last few years.

Another technique, called *TOPP*, for measuring avail-bw was proposed in [59]. The key idea in TOPP is: the ratio of input to output probing rate is smaller than one if the input probing rate is greater than avail-bw, otherwise it is equal to one. TOPP uses sequences of packet pairs sent to the path at increasing rates. From the relation between the input and output rates of different packet pairs, it is possible to estimate the avail-bw and the capacity of the tight link in the path. In certain path configurations, it is also possible to measure the avail-bw and capacity of other links in the path. However, the TOPP approach has one major limitation. TOPP is basically a methodology, which shows the feasibility of key idea described earlier for avail-bw estimation, and not a complete algorithm to estimate avail-bw. In a network path, output probing rate is affected by number of factors, in addition to tight link avail-bw, such as interference from cross traffic packets at non-tight links, burstiness of cross traffic, etc. TOPP methodology does not include an algorithm to measure the avail-bw in real network path. Additionally, its approach of using the relation between the input and the output rates, however, introduces errors in th estimation. We will return to the previous points in chapter 7.

1.2 Contributions

The key contributions of this thesis are:

- Establishing the precise definition of the avail-bw and designing SLoPS methodology for avail-bw estimation.
- Design and development of tool, called Pathload, for estimating average avail-bw.
- Formulating avail-bw as a random process and developing two algorithms to measure the variability of avail-bw process. Implemented these algorithms in a tool called Pathvar.
- Study of effect of Interrupt Coalescence on the accuracy of bandwidth estimation tools and design of an algorithm to enable bandwidth estimation tool to function accurately in presence of IC.

- Study of the variability of avail-bw process.
- Demonstrate the performance improvements achieved in video streaming by using avail-bw as opposed to simpler metrics such as network loss rate and delay.

1.2.1 Avail-bw Estimation Methodology: *SLoPS*

Our first contribution is an e2e methodology, called Self-Loading Periodic Streams (SLoPS), for measuring avail-bw [33]. The basic idea in SLoPS is that the one-way delays of a periodic packet stream show an increasing trend when the stream's rate is higher than the avail-bw. SLoPS has numerous applications, such as server selection, routing in overlay networks, rate adaptation in video streaming, and we show one such application of SLoPS.

1.2.2 Average Available Bandwidth: Algorithm and Tool

We have implemented SLoPS in a tool called *Pathload* [33]. Pathload implements an iterative algorithm to estimate the average avail-bw [33]. We have evaluated the accuracy of the tool with both simulations and experiments over real-world Internet paths. Pathload has also been verified and shown to be more accurate than other tools by independent researchers [80]. We show that Pathload is non-intrusive, meaning that it does not cause a significant increase in the network utilization, average delays, or loss rate.

1.2.3 Available Bandwidth Variation Range: Algorithm and Tool

Next, we view the avail-bw as a stationary random process. We focus on the e2e estimation of the variability of the avail-bw marginal distribution. Specifically, we develop two techniques to estimate the percentiles of avail-bw distribution. The first is iterative and non-parametric, meaning that it is more appropriate for very short time scales (typically less than 100ms), or in bottlenecks with limited flow multiplexing (where the avail-bw distribution may be non-Gaussian). The second technique is parametric, because it assumes that the avail-bw follows the Gaussian distribution, and it can produce an estimate faster because it is not iterative. The two techniques have been implemented in a measurement tool called *Pathvar* [35]. Pathvar can track the avail-bw variation range within 10%-20%, even under non-stationary conditions. Finally, we identify four factors that play a crucial

role in the variation range of the avail-bw: traffic load, number of competing flows, rate of competing flows, and of course the measurement time scale.

1.2.4 Effect of Interrupt Coalescence on Bandwidth Estimation Tools

The accuracy of bandwidth estimation tools can be affected by several factors which are beyond the control of measurement tools. We show [72] that Interrupt Coalescence (IC), a technique implemented in network cards to reduce interrupt overhead, is one such factor affecting the accuracy of most avail-bw estimation tools. We first explain how IC works in two popular Gigabit Ethernet controllers. Then, we identify the negative effects of IC on active and passive network measurements. Specifically, we show that IC can affect active bandwidth estimation techniques, causing erroneous measurements. It can also alter the packet interarrivals in passive monitors that use commodity network interfaces. Based on the "signature" that IC leaves on the dispersion and one-way delays of packet trains, we show how to detect IC and how to filter its effects from raw measurements.

1.2.5 Misconceptions in Available Bandwidth

We review several avail-bw estimation techniques and tools developed in last 4-5 years [34], mostly after the publication of Pathload. We show that some key issues regarding the avail-bw definition, estimation, and validation remain vague or misinterpreted. We then identify ten misconceptions and clarify them using analysis and/or simulations to enhance the understanding of avail-bw as a useful metric.

1.2.6 Video Streaming using Avail-bw

Finally, we present a adaptive path selection approach using avail-bw estimation to maximize perceived quality of streaming video. Selecting one of several Internet paths is currently possible using multihoming and/or overlay routing infrastructures. Specifically, we conduct an experimental comparison of various measurement-based path selection techniques for video streaming. The path selection is based on the measurement of network-layer metrics, such as loss rate, jitter or available bandwidth, while the video quality is evaluated based on the VQM tool. Our experiments show that the most effective technique for adaptive

path selection relies on the inference of a lower bound of the available bandwidth. We also show how to perform such measurements using video packets, eliminating the measurement overhead in the selected path.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 describes the SLoPS methodology that infers whether the available bandwidth is greater or smaller than a given probing rate. Chapter 3 presents a tool, called Pathload, to estimate the average available bandwidth. Chapter 4 presents a tool called Pathvar to estimate the variation range of avail-bw. In Chapter 5, we present the effect of interrupt coalescence on the accuracy of bandwidth estimation tools. Chapter 6 presents an application of avail-bw estimation to video streaming in the Internet. Chapter 7 discusses other recent bandwidth estimation proposals and clarifies some misconceptions in the area of available bandwidth estimation. Finally, chapter 8 presents a summary of the contributions of this thesis and some issues that deserve further research.

CHAPTER II

SELF LOADING PERIODIC STREAM (SLOPS)

In this chapter, we first define the avail-bw in an intuitive but precise manner. The definition does not depend on higher-level issues, such as the transport protocol or the number of flows that can capture the avail-bw in a path. Next, we present an original end-to-end avail-bw measurement methodology, called *Self-Loading Periodic Streams* or *SLoPS*. The basic idea in SLoPS is that the one-way delays of a periodic packet stream show an increasing trend when the stream's rate is higher than the avail-bw. SLoPS methodology is the core of two avail-bw estimation tools that are presented in chapter 3 and 4.

2.1 Definitions

A network path $\mathcal{P}$ is a sequence of H store-and-forward links that transfer packets from a sender $\mathcal{SND}$ to a receiver $\mathcal{RCV}$. We assume that the path is fixed and unique, i.e., no routing changes or multipath forwarding occur during the measurements. Each link i can transmit data with a rate C_i bps, that is referred to as *link capacity*. Two throughput metrics that are commonly associated with $\mathcal{P}$ are the end-to-end *capacity* C and *available bandwidth* A. The capacity is defined as

$$C \equiv \min_{i=1...H} C_i \tag{1}$$

and it is *the maximum rate that the path can provide to a flow, when there is no other traffic in* $\mathcal{P}$.

Suppose that link i transmitted $C_i u_i(t_0, t_0 + \tau)$ bits during a time interval $(t_0, t_0 + \tau)$. The term $u_i(t_0, t_0+\tau)$, or simply $u_i^\tau(t_0)$, is the average *utilization* of link i during $(t_0, t_0+\tau)$, with $0 \leq u_i^\tau(t_0) \leq 1$. Intuitively, the avail-bw $A_i^\tau(t_0)$ of link i in $(t_0, t_0 + \tau)$ can be defined as the fraction of the link's capacity that has not been utilized during that interval, i.e.,

$$A_i^\tau(t_0) \equiv C_i \left[1 - u_i^\tau(t_0)\right] \tag{2}$$

Extending this concept to the entire path, the end-to-end avail-bw $A^\tau(t_0)$ during $(t_0, t_0+\tau)$ is the minimum avail-bw among all links in $\mathcal{P}$,

$$A^\tau(t_0) \equiv \min_{i=1\ldots H} \{C_i \left[1 - u_i^\tau(t_0)\right]\} \tag{3}$$

Thus, *the end-to-end avail-bw is defined as the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic in* $\mathcal{P}$.

To avoid the term *bottleneck link*, that has been widely used in the context of both capacity and avail-bw, we introduce two new terms. The *narrow link* is the link with the minimum capacity, and it determines the capacity of the path. The *tight link*, on the other hand, is the link with the minimum avail-bw, and it determines the avail-bw of the path.

The parameter τ in (3) is the avail-bw *averaging timescale*. If we consider $A^\tau(t)$ as a stationary random process, the variance $\mathrm{Var}\{A^\tau\}$ of the process decreases as the averaging timescale τ increases. We note that if A^τ is self-similar, the variance $\mathrm{Var}\{A^\tau\}$ *decreases slowly*, in the sense that the decrease of $\mathrm{Var}\{A^\tau\}$ as τ increases is slower than the reciprocal of τ [53].

2.2 Self-Loading Periodic Streams

In this section, we describe the Self-Loading Periodic Streams (*SLoPS*) measurement methodology. A periodic stream in SLoPS consists of K packets of size L, sent to the path at a constant rate R. *If the stream rate R is higher than the avail-bw A, the one-way delays of successive packets at the receiver show an increasing trend.* We first illustrate this fundamental effect in its simplest form through an analytical model with stationary and fluid cross traffic. Then, we show how to use this 'increasing delays' property in an iterative algorithm that measures end-to-end avail-bw. Finally, we depart from the previous fluid model, and observe that the avail-bw may vary during a stream. This requires us to refine SLoPS in several ways, that is the subject of the next section.

2.2.1 SLoPS with fluid cross traffic

Consider a path from $\mathcal{SND}$ to $\mathcal{RCV}$ that consists of H links, $i = 1, \ldots, H$. The capacity of link i is C_i. We consider a stationary (i.e., time invariant) and fluid model for the cross traffic

in the path. So, if the avail-bw at link i is A_i, the utilization is $u_i = (C_i - A_i)/C_i$ and there are $u_i\, C_i\, \tau$ bytes of cross traffic departing from, and arriving at, link i in any interval of length τ. Also, assume that the links follow the First-Come First-Served queueing discipline[1], and that they are adequately buffered to avoid losses. We ignore any propagation or fixed delays in the path, as they do not affect the delay variation between packets. The avail-bw A in the path is determined by the tight link $t \in \{1, \ldots, H\}$ with[2]

$$A_t = \min_{i=1\ldots H} A_i = \min_{i=1\ldots H} C_i\,(1 - u_i) = A \tag{4}$$

Suppose that $\mathcal{SND}$ sends a periodic stream of K packets to $\mathcal{RCV}$ at a rate R_0, starting at an arbitrary time instant. The packet size is L bytes, and so packets are sent with a period of $T = L/R_0$ time units. The One-Way Delay (OWD) D^k from $\mathcal{SND}$ to $\mathcal{RCV}$ of packet k is

$$D^k = \sum_{i=1}^{H} (\frac{L}{C_i} + \frac{q_i^k}{C_i}) = \sum_{i=1}^{H} (\frac{L}{C_i} + d_i^k) \tag{5}$$

where q_i^k is the queue size at link i upon the arrival of packet k (q_i^k does not include packet k), and $d_i^k = q_i^k/C_i$ is the queueing delay of packet k at link i. The OWD difference between two successive packets k and $k+1$ is

$$\Delta D^k \equiv D^{k+1} - D^k = \sum_{i=1}^{H} \frac{\Delta q_i^k}{C_i} = \sum_{i=1}^{H} \Delta d_i^k \tag{6}$$

where $\Delta q_i^k \equiv q_i^{k+1} - q_i^k$, and $\Delta d_i^k \equiv \Delta q_i^k / C_i$.

We can now show that, if $R_0 > A$ the K packets of the periodic stream will arrive at $\mathcal{RCV}$ with increasing OWDs, while if $R_0 \leq A$ the stream packets will encounter equal OWDs. This property is stated next with the proof.

Proposition 1 *If $R_0 > A$, then $\Delta D^k > 0$ for $k = 1, \ldots K-1$. Else, if $R_0 \leq A$, $\Delta D^k = 0$ for $k = 1, \ldots K-1$.*

Proof:

[1] In links with per-flow or per-class queueing, SLoPS can obviously monitor the sequence of queues that the probing packets go through.

[2] If there are more than one links with avail-bw A, the tight link is the first of them, without loss of generality.

At the first link

Case 1: $R_0 > A_1$.

Suppose that t^k is the arrival time of packet k in the queue. Over the interval $[t^k, t^k + T)$, with $T = L/R_0$, the link is constantly backlogged because the arriving rate is higher than the capacity ($R_0 + u_1\, C_1 = C_1 + (R_0 - A_1) > C_1$). Over the same interval, the link receives $L + u_1\, C_1\, T$ bytes and services $C_1\, T$ bytes. Thus,

$$\Delta q_1^k = (L + u_1\, C_1\, T) - C_1\, T = (R_0 - A_1)\, T > 0 \tag{7}$$

and so,

$$\Delta d_1^k = \frac{R_0 - A_1}{C_1}\, T > 0. \tag{8}$$

Packet $k+1$ departs the first link Λ time units after packet k, where

$$\Lambda = (t^{k+1} + d_1^{k+1}) - (t^k + d_1^k) = T + \frac{R_0 - A_1}{C_1}\, T \tag{9}$$

that is independent of k. So, *the packets of the stream depart the first link with a constant rate* R_1, where

$$R_1 = \frac{L}{\Lambda} = R_0 \frac{C_1}{C_1 + (R_0 - A_1)}. \tag{10}$$

We refer to rate R_{i-1} as the *entry-rate* in link i, and to R_i as the *exit-rate* from link i. Given that $R_0 > A_1$ and that $C_1 \geq A_1$, it is easy to show that *the exit-rate from link 1 is larger or equal than* A_1[3] *and lower than the entry-rate*,

$$A_1 \leq R_1 < R_0. \tag{11}$$

Case 2: $R_0 \leq A_1$.

In this case, the arrival rate at the link in interval $[t^k, t^k + T)$ is $R_0 + u_1\, C_1 \leq C_1$. So, packet k is serviced before packet $k+1$ arrives at the queue. Thus,

$$\Delta q_1^k = 0, \quad \Delta d_1^k = 0, \text{ and } \; R_1 = R_0 \tag{12}$$

Induction to subsequent links The results that were previously derived for the first link can be proved inductively for each link in the path. So, we have the following relationship

[3] $R_1 = A_1$ *when* $A_1 = C_1$.

between the entry and exit rates of link i:

$$R_i = \begin{cases} R_{i-1}\frac{C_i}{C_i+(R_{i-1}-A_i)} & \text{if } R_{i-1} > A_i \\ R_{i-1} & \text{otherwise} \end{cases} \tag{13}$$

with

$$A_i \leq R_i < R_{i-1} \quad \text{when } R_{i-1} > A_i \tag{14}$$

Consequently, the exit-rate from link i is

$$R_i \geq \min\{R_{i-1}, A_i\} \tag{15}$$

Also, the queueing delay difference between successive packets at link i is

$$\Delta d_i^k = \begin{cases} \frac{R_{i-1}-A_i}{C_i}\,T > 0 & \text{if } R_{i-1} > A_i \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

OWD variations If $R_0 > A$, we can apply (14) recursively for $i = 1, \ldots (t-1)$ to show that the stream will arrive at the tight link with a rate $R_{t-1} \geq A_{t-1} > A_t$. Thus, based on (16), $\Delta d_t^k > 0$, and so the OWD difference between successive packets will be positive, $\Delta d^k > 0$.

On the other hand, if $R_0 \leq A$, then $R_0 \leq A_i$ for every link i (from the definition of A). So, applying (15) recursively from the first link to the last, we see that $R_i < A_i$ for $i = 1, \ldots H$. Thus, (16) shows that the delay difference in each link i is $\Delta d_i^k = 0$, and so the OWD differences are $\Delta d^k = 0$.

One may think that the avail-bw A can be computed directly from the rate at which the stream arrives at $\mathcal{RCV}$. This is the approach followed in packet train dispersion techniques. The following result, however, shows that, in a general path configuration, this would be possible only if the capacity and avail-bw of all links (except the avail-bw of the tight link) are *a priori* known.

Proposition 2 *The rate R_H of the packet stream at $\mathcal{RCV}$ is a function, in the general case, of C_i and A_i for all $i = 1, \ldots, H$.*

This result follows from the proof of proposition 1 (apply recursively Equation 13 until $i = H$).

2.2.2 An iterative algorithm to measure A

Based on Proposition 1, we can construct an iterative algorithm for the end-to-end measurement of A. Suppose that $\mathcal{SND}$ sends a periodic stream n with rate $R(n)$. The receiver analyzes the OWD variations of the stream, based on Proposition 1, to determine whether $R(n) > A$ or not. Then, $\mathcal{RCV}$ notifies $\mathcal{SND}$ about the relation between $R(n)$ and A. If $R(n) > A$, $\mathcal{SND}$ sends the next periodic stream $n+1$ with rate $R(n+1) < R(n)$. Otherwise, the rate of stream $n+1$ is $R(n+1) > R(n)$.

Specifically, $R(n+1)$ can be computed as follows,

$$\begin{aligned} \text{If } R(n) > A, R^{max} &= R(n); \\ \text{If } R(n) \leq A, R^{min} &= R(n); \\ R(n+1) &= (R^{max} + R^{min})/2; \end{aligned} \tag{17}$$

R^{min} and R^{max} are lower and upper bounds for the avail-bw after stream n, respectively. Initially, R^{min}=0 and R^{max} can be set to a sufficiently high value $R_0^{max} > A$.[4] The algorithm terminates when $R^{max} - R^{min} \leq \omega$, where ω is the user-specified *estimation resolution*. If the avail-bw A does not vary with time, the previous algorithm will converge to a range $[R^{min}, R^{max}]$ that includes A after $\lceil \log_2(R_0^{max}/\omega) \rceil$ streams.

2.2.3 SLoPS with real cross traffic

We assumed so far that the avail-bw A is constant during the measurement process. In reality, the avail-bw may vary because of two reasons. First, the avail-bw process $A^\tau(t)$ of (3) may be non-stationary, and so its expected value may also be a function of time. Even if $A^\tau(t)$ is stationary, however, the process A^τ can have a significant statistical variability around its (constant) mean $E[A^\tau]$, and to make things worse, this variability may extend over a wide range of timescales τ. How can we refine SLoPS to deal with the dynamic nature of the avail-bw process?

To gain some insight into this issue, Figures 2, 3, and 4 show the OWD variations in three periodic streams that crossed a 12-hop path from Univ-Oregon to Univ-Delaware. All

[4] A better way to initialize R^{max} is described in chapter 3.

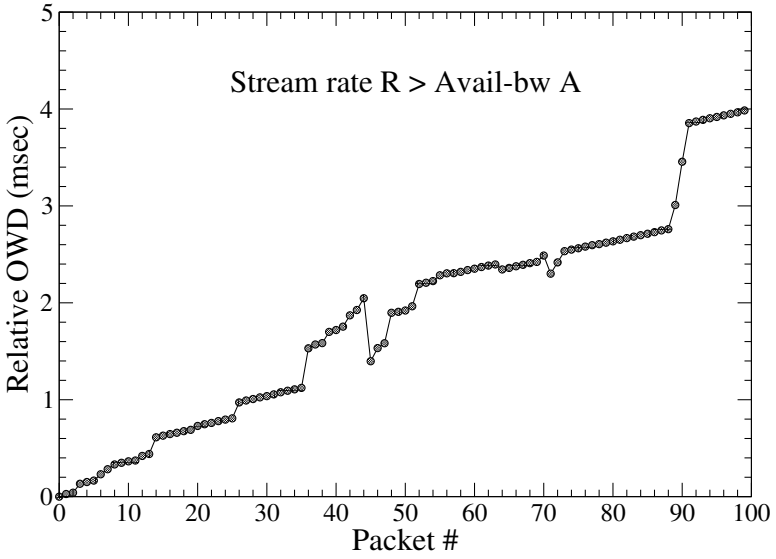


Figure 2: OWD variations for a periodic stream when $R > A$.

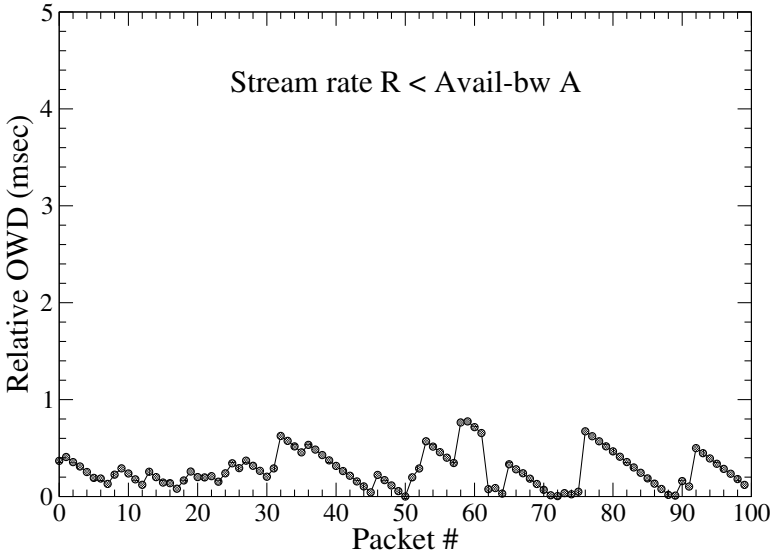


Figure 3: OWD variations for a periodic stream when $R < A$.

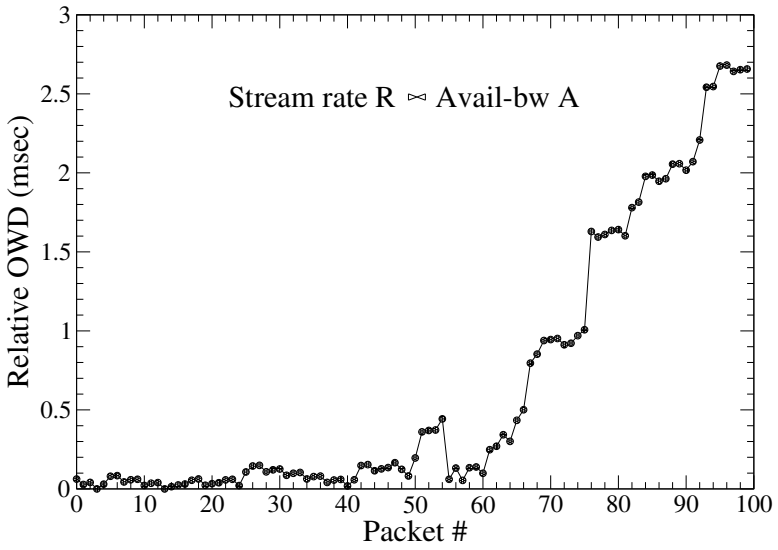


Figure 4: OWD variations for a periodic stream when $R \bowtie A$.

three streams have K=100 packets with T=100μs. The 5-minute average avail-bw in the path during these measurements was about 74Mbps, according to the MRTG utilization graph of the tight link. In Figure 2, the stream rate is R=96Mbps, i.e., higher than the long-term avail-bw. Notice that the OWDs between successive packets are not strictly increasing, as one would expect from Proposition 1, but *overall, the stream OWDs have a clearly increasing trend*. This is shown both by the fact that most packets have a higher OWD than their predecessors, and because the OWD of the last packet is about 4ms larger than the OWD of the first packet. On the other hand, the stream of Figure 3 has a rate R=37Mbps, i.e., lower than the long-term avail-bw. Even though there are short-term intervals in which we observe increasing OWDs, *there is clearly not an increasing trend in the stream*. The third stream, in Figure 4, has a rate R=82Mbps. The stream does not show an increasing trend in the first half, indicating that the avail-bw during that interval is higher than R. The situation changes dramatically, however, after roughly the 60-th packet. In that second half of the stream there is a clear increasing trend, showing that the avail-bw decreases to less than R.

The previous example motivates two important refinements in the SLoPS methodology. First, instead of analyzing the OWD variations of a stream, expecting one of the two cases

of Proposition 1 to be strictly true for every pair of packets, we should instead watch for the presence of *an overall increasing trend during the entire stream*. Second, we have to accept the possibility that the avail-bw may vary around rate R during a probing stream. In that case, there is no strict ordering between R and A, and thus a third possibility comes up, that we refer to as 'grey-region' (denoted as $R \bowtie A$).

2.3 Summary

In this chapter, we described an orignal end-to-end available bandwidth measurement methodology, $SLoPS$, based on the idea that one-way delays of a periodic stream show an increasing trend if the stream rate is greater than avail-bw. An end-to-end avail-bw measurement methodology, such as SLoPS, can have numerous applications, such as bandwidth-Delay-Product in TCP, overlay networks and end-system multicast, rate adaptation in streaming applications, end-to-end admission control, server selection and anycasting, and verification of SLAs. In the next chapter, we present design of average avail-bw estimation tool called Pathload.

CHAPTER III

AVERAGE AVAIL-BW ESTIMATION: PATHLOAD

In this chapter, we present the algorithm design and its implementation in a tool, called *Pathload*, to estimate average avail-bw. An important feature of *Pathload* is that, instead of reporting a single figure for the average avail-bw in a time interval $(t_0, t_0 + \Theta)$, *it estimates the range in which the avail-bw process* $A^\tau(t)$ *varies in* $(t_0, t_0 + \Theta)$*, when it is measured with an averaging timescale* $\tau < \Theta$. The timescales τ and Θ are related to two tool parameters, namely the 'stream duration' and the 'fleet duration'.

Pathload has been verified experimentally, by comparing its results with MRTG utilization graphs for the path links [65]. We have also evaluated *Pathload* in a controlled and reproducible environment using NS simulations. The simulations show that *Pathload* reports a range that includes the average avail-bw in a wide range of load conditions and path configurations. The tool underestimates the avail-bw, however, when the path includes several tight links. The *Pathload* measurements are non-intrusive, meaning that they do not cause significant increases in the network utilization, delays, or losses.

We have used *Pathload* to estimate the variability (or 'dynamics') of the avail-bw in different paths and load conditions. An important observation is that the avail-bw becomes more variable as the utilization of the tight link increases (i.e., as the avail-bw decreases). Similar observations are made for paths of different capacity that operate at about the same utilization. Specifically, the avail-bw shows higher variability in paths with smaller capacity, probably due to a lower degree of statistical multiplexing.

Finally, we examined the relation between the avail-bw and the throughput of a 'greedy' TCP connection, i.e., a persistent bulk transfer with sufficiently large advertised window. Our experiments show that such a greedy TCP connection can be used to roughly measure the end-to-end avail-bw, but TCP saturates the path, increases significantly the delays and jitter, and potentially causes losses to other TCP flows. The increased delays and losses in

the path cause other TCP flows to slow down, allowing the greedy TCP connection to grab more bandwidth than what was previously available.

3.1 Measurement tool: Pathload

We implemented SLoPS in a tool called *Pathload*. *Pathload* consists of two components: process $\mathcal{SND}$ running at the sender and process $\mathcal{RCV}$ running at the receiver. The tool uses UDP for the periodic packet streams. Additionally, a TCP connection between the two end-points serves as a 'control channel'. The control channel transfers messages regarding the characteristics of each stream, the abortion or end of the measurement process, etc. In the following, we describe *Pathload* in detail.

3.1.1 Clock issues

$\mathcal{SND}$ timestamps each packet upon its transmission. So, $\mathcal{RCV}$ can measure the *relative OWD* D^k of packet k, that differs from the actual OWD by a certain offset. This offset is due to the non-synchronized clocks of the end-hosts. Since we are only interested in OWD differences though, a constant offset in the measured OWDs does not affect the analysis. Clock skew can be a potential problem,[1] but not in *Pathload*. The reason is that each stream lasts for only a few milliseconds (§3.1.2), and so the skew during a stream is in the order of nanoseconds, much less than the OWD variations due to queueing.

3.1.2 Stream parameters

A stream consists of K packets of size L, sent at a constant rate R. The reason for sending a stream with constant rate is: stream duration is related to the averaging timescale, which corresponds to the duration over which the relationship between rate R and avail-bw A is examined. R is adjusted at run-time for each stream, as described in §3.1.5. The packet inter-spacing T is normally set to T_{min}, which is based on the minimum possible period that the end-hosts can achieve. The packet inter-spacing is constant for all the packets in the stream to achieve a constant rate R. Given R and T, the packet size is computed as $L = RT$. L, however, has to be smaller than the path MTU L^{max} (to avoid fragmentation),

[1] And there are algorithms to remove its effects [69].

and larger than a minimum possible size L^{min}=200B. The reason for the L^{min} constraint is to reduce the effect of Layer-2 headers on the stream rate (see [68]). If $R < L^{min}/T_{min}$, the inter-spacing T is increased to L^{min}/R. The maximum rate that *Pathload* can generate, and thus the maximum avail-bw that it can measure, is L^{max}/T_{min}.

The stream length K is chosen based on two constraints. First, a stream should be relatively short, so that it does not cause large queues and potentially losses in the path routers. Second, K controls the *stream duration* $V = KT$, which is related to the *averaging timescale* τ (see §3.3.3). A larger K (longer stream) increases τ, and thus reduces the variability in the measured avail-bw. In *Pathload*, the default value for K is 100 packets.

3.1.3 Detecting an increasing OWD trend

Suppose that the (relative) OWDs of a particular stream are $D^1, D^2, \ldots, D^K$. As a pre-processing step, we partition these measurements into $\Gamma = \sqrt{K}$ groups of Γ consecutive OWDs. Then, we compute the median OWD $\hat{D}^k$ of each group. *Pathload* analyzes the set $\{\hat{D}^k, k = 1, \ldots, \Gamma\}$, which is more robust to outliers and errors.

We use two complementary statistics to check if a stream shows an increasing trend. The *Pairwise Comparison Test* (PCT) metric of a stream is

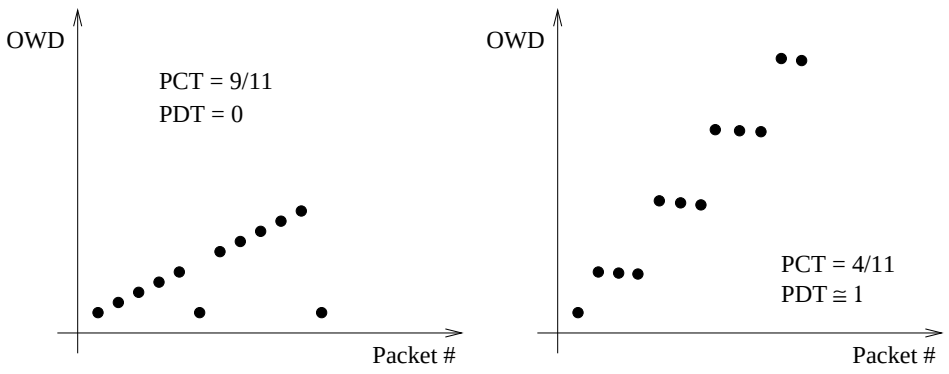
$$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}^k > \hat{D}^{k-1})}{\Gamma - 1} \tag{18}$$

where $I(X)$ is one if X holds, and zero otherwise. PCT measures the fraction of consecutive OWD pairs that are increasing, and so $0 \leq S_{PCT} \leq 1$. If the OWDs are independent, the expected value of S_{PCT} is 0.5. If there is a strong increasing trend, S_{PCT} approaches one.

The *Pairwise Difference Test* (PDT) metric of a stream is

$$S_{PDT} = \frac{\hat{D}^{\Gamma} - \hat{D}^1}{\sum_{k=2}^{\Gamma} |\hat{D}^k - \hat{D}^{k-1}|} \tag{19}$$

PDT quantifies how strong is the start-to-end OWD variation, relative to the OWD absolute variations during the stream. Note that $-1 \leq S_{PDT} \leq 1$. If the OWDs are independent, the expected value of S_{PDT} is zero. If there is a strong increasing trend, S_{PDT} approaches one.



(a) High PCT - Low PDT (b) Low PCT - High PDT

Figure 5: Comparison of PCT and PDT for a stream with increasing OWD trend.

There are cases in which one of the two metrics is better than the other in detecting an increasing trend. For instance, Figure 5 shows two 12-packet streams with an increasing OWD trend. In Figure 5-a, PCT would detect the increasing trend but PDT would not, while the opposite would be the case in Figure 5-b.

In *Pathload*, we characterize the trend in a stream as follows. When one of the PCT and PDT metrics reports 'increasing trend', while the other is either 'increasing' or 'ambiguous', the stream is characterized as *type-I* (for 'increasing'). Similarly, when one metric reports 'non-increasing' trend, while the other is either 'non-increasing' or 'ambiguous', the stream is characterized as *type-N* (for 'non-increasing'). If both metrics report 'ambiguous', or when one is 'increasing' and the other is 'non-increasing', the stream is discarded. We show the sensitivity of Pathload results to PDT threshold in the next section.

3.1.4 Fleets of streams

Pathload does *not* determine whether $R > A$ based on a single stream. Instead, it sends a *fleet* of N streams, so that it samples whether $R > A$ N successive times. All streams in a fleet have the same rate R. Each stream is sent only when the previous stream has been acknowledged, to avoid a backlog of streams in the path. So, there is always an idle interval Δ between streams, which is larger than the Round-Trip Time (RTT) of the path. The duration of a fleet is $U{=}N(V + \Delta){=}N(KT + \Delta)$. Given V and Δ, N determines the

fleet duration, which is related to the *Pathload* measurement latency. The default value for N is 12 streams. The effect of N is discussed in §3.3.4.

The average *Pathload* rate during a fleet of rate R is

$$\frac{NKL}{N(V+\Delta)} = R\frac{1}{1+\frac{\Delta}{V}}$$

In order to limit the average *Pathload* rate to less than 10% of R, the current version of *Pathload* sets the inter-stream latency to $\Delta = \max\{RTT, 9\,V\}$.

If a stream encounters excessive losses (>10%), or if more than a number of streams within a fleet encounter moderate losses (>3%), the entire fleet is aborted and the rate of the next fleet is decreased.

Grey-region If a large fraction f of the N streams in a fleet are of type-I, the entire fleet shows an increasing trend and we infer that the fleet rate is larger than the avail-bw ($R > A$). Similarly, if a fraction f of the N streams are of type-N, the fleet does not show an increasing trend and we infer that the fleet rate is smaller than the avail-bw ($R < A$). It can happen, though, that less than $N \times f$ streams are of type-I, and also that less than $N \times f$ streams are of type-N. In that case, some streams 'sampled' the path when the avail-bw was less than R (type-I), and some others when it was more than R (type-N). We say, then, that the fleet rate R is in the 'grey-region' of the avail-bw, and write $R \bowtie A$. The interpretation that we give to the grey-region is that *when $R \bowtie A$, the avail-bw process $A^\tau(t)$ during that fleet varied above and below rate R, causing some streams to be of type-I and some others to be of type-N.* The averaging timescale τ, here, is related to the stream duration V. We discuss the effect of f on *Pathload* results in the next section.

The previous three cases are illustrated in Figures 6 and 7. Figure 6 shows the PCTs for three fleets (with N=20 streams), while Figure 7 shows the PDTs for the same fleets. In the first fleet (R=94Mbps, A=75Mbps), all streams are characterized as type-I. In the second fleet (R=80Mbps, A=74Mbps), 11 streams are type-I, 6 are type-N, and 3 are discarded. So, in this case, the rate R is in the grey-region of the avail-bw ($R \bowtie A$). In the third fleet (R=40Mbps, A=74Mbps), all streams are characterized as type-N.

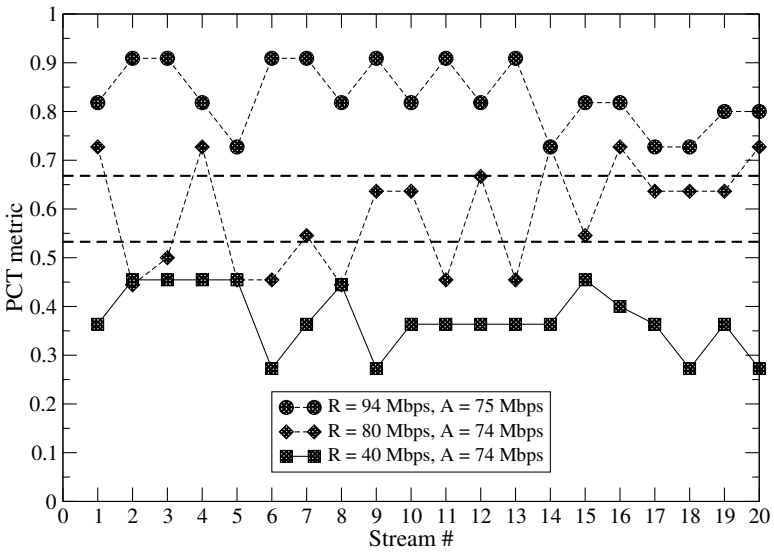


Figure 6: PCTs for three fleets (20 streams each).

3.1.5 Rate adjustment algorithm

After a fleet n of rate $R(n)$ is over, *Pathload* determines whether $R(n) > A$, $R(n) < A$, or $R(n) \bowtie A$. The iterative algorithm that determines the rate $R(n+1)$ of the next fleet is quite similar to the binary-search approach of (17). There are two important differences though.

First, together with the upper and lower bounds for the avail-bw R^{max} and R^{min}, *Pathload* also maintains upper and lower bounds for the grey-region, namely G^{max} and G^{min}. When $R(n) \bowtie A$, one of these bounds is updated depending on whether $G^{max} < R(n) < R^{max}$ (update G^{max}), or $G^{min} > R(n) > R^{min}$ (update G^{min}). If a grey-region has not been detected up to that point, the next rate $R(n+1)$ is chosen, as in (17), half-way between R^{min} and R^{max}. If a grey-region has been detected, $R(n+1)$ is set half-way between G^{max} and R^{max} when $R(n) = G^{max}$, or half-way between G^{min} and R^{min} when $R(n) = G^{min}$. It is important to note that this binary-search approach succeeds in converging to the avail-bw, as long as the avail-bw variation range is strictly included in the $[R^{min}, R^{max}]$ range. The experimental and simulation results of the next section show that this is the case generally, with the exception of paths that include several tight links.

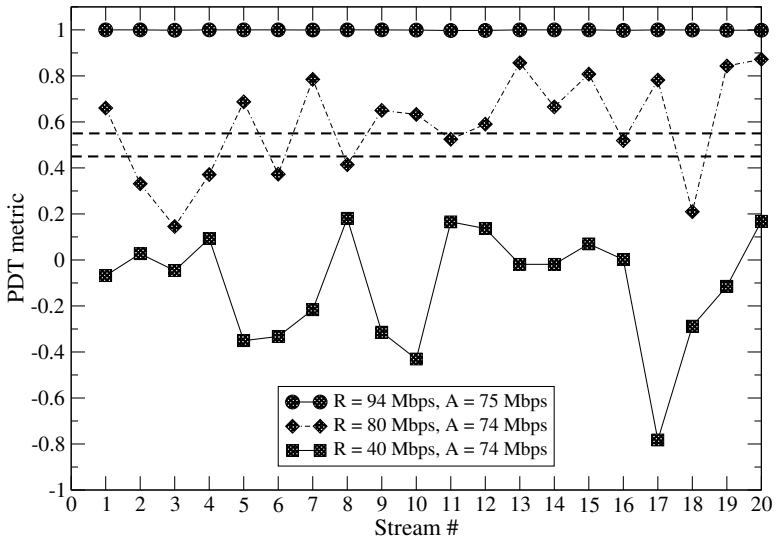


Figure 7: PDTs for three fleets (20 streams each).

The second difference is that *Pathload* terminates not only when the avail-bw has been estimated within a certain resolution ω (i.e., $R^{max} - R^{min} \leq \omega$), but also when $R^{max} - G^{max} \leq \chi$ and $G^{min} - R^{min} \leq \chi$, i.e., when both avail-bw boundaries are within χ from the corresponding grey-region boundaries. The parameter χ is referred to as *grey-region resolution*.

The tool eventually reports the range $[R^{min}, R^{max}]$.

3.1.6 Measurement latency

Since *Pathload* is based on an iterative algorithm, it is hard to predict how long will a measurement take. For the default tool parameters, and for a path with $A \approx$ 100Mbps and Δ=100ms, the tool needs less than 15 seconds to produce a final estimate. The measurement latency increases as the absolute magnitude of the avail-bw and/or the width of the grey-region increase, and it also depends on the resolution parameters ω and χ.

3.1.7 Response to packet losses

After each stream is received, $\mathcal{RCV}$ measures the loss rate that the stream experienced. If a stream encountered moderate losses (<3%), the stream is marked as lossy. If more than a certain fraction of streams have been marked as lossy, then $\mathcal{RCV}$ informs $\mathcal{SND}$ to abort

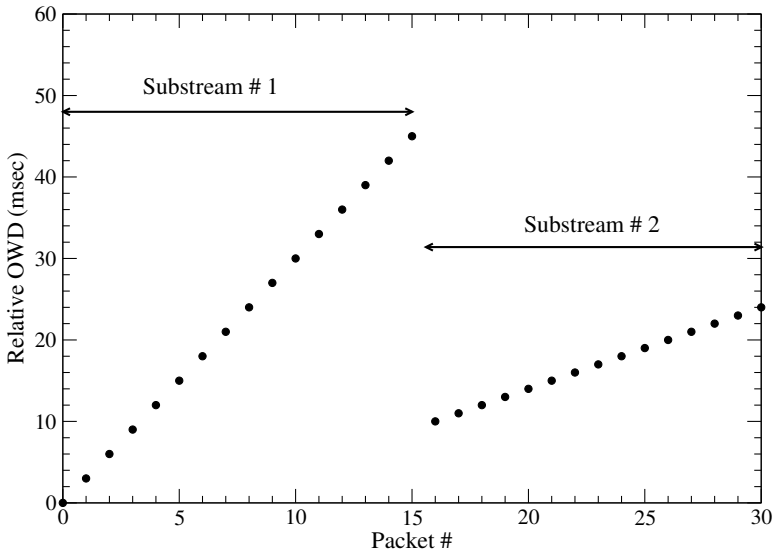


Figure 8: A context switch at the sender.

the remaining streams of the fleet. Also, if a stream encountered excessive losses ($> 10\%$), the fleet is immediately aborted. The rate R at which the fleet was aborted becomes the new upper bound R^{max} of the rate adjustment algorithm.

3.1.8 Detection of a sender context switch

Pathload checks whether a context switch occurred at $\mathcal{SND}$ while a stream was being sent. Suppose that t_i is the transmission time of packet i from $\mathcal{SND}$. t_i is carried in packet i. $\mathcal{RCV}$ compares the sending times of consecutive packets to see whether $t_{i+1} - t_i > T + W$, where W is maximum allowed deviation from the target period T. If $t_{i+1} - t_i > T + W$, *Pathload* takes the 'pessimistic' approach that $\mathcal{SND}$ was switched out after sending the i^{th} packet of a stream. Then, $\mathcal{RCV}$ splits the received stream into two substreams, one between packets 1 and i, and another between packets $i+1$ and K. If a substream includes less than $K/2$ packets, it is discarded from the OWD analysis. In the current version of *Pathload*, W is set to 10ms.

Figure 8 shows an illustration of a 30-packet stream in which a context switch occurred at $\mathcal{SND}$. Packets 1-15 are sent before $\mathcal{SND}$ is switched out, while the remaining 15 packets are sent after it runs again. Both substreams show a increasing trend, indicating that a queue was building up at the tight link. In this example, we assume that the avail-bw

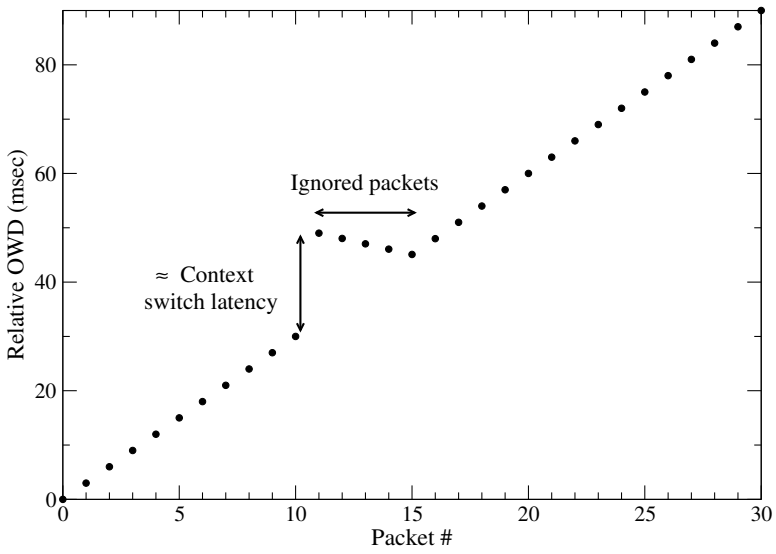


Figure 9: A context switch at the receiver.

changed between the two substreams, causing different increasing slopes in the OWDs of the two substreams.

3.1.9 Detection of a receiver context switch

Pathload also checks whether a context switch occurred at $\mathcal{RCV}$ while a stream was being received. Suppose that a_i is the arrival time of packet i at the $\mathcal{RCV}$ *Pathload* process. If $\mathcal{RCV}$ is switched out while receiving a stream, some of the stream packets will be accumulated in a kernel buffer at the receiving host. When $\mathcal{RCV}$ runs again, those packets are transferred from kernel to user space with a spacing of Q μs, where Q is the latency of the *recvfrom* system call. Typically, Q is a few microseconds, and it can be measured at $\mathcal{RCV}$ before the measurements start. So, $\mathcal{RCV}$ can detect a local context switch comparing the arrival times of consecutive packets. If $a_{i+1} - a_i \approx Q$, packets i and $i+1$ are marked as backlogged at the kernel, and they are discarded from the OWD analysis.

Figure 9 shows an illustration of a 30-packet stream in which a context switch occurred at the receiver. Based on the arrival timestamps a_i, $\mathcal{RCV}$ determines that the context switch affected packets 11-14, and discards them from the OWD analysis. Packet 15 and onwards arrive at the receiver after $\mathcal{RCV}$ started running again. Hence, their OWD measurements are not affected by the previous context switch. Notice that we do not need to analyze

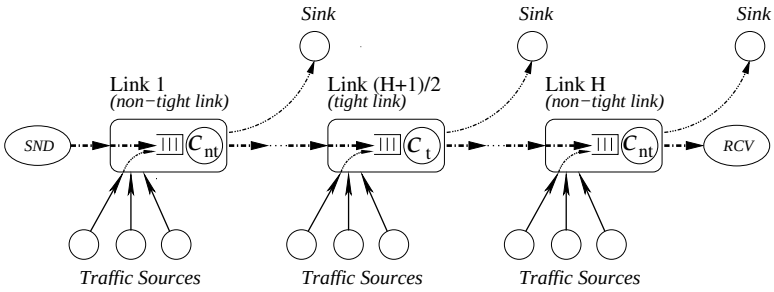


Figure 10: Simulation topology.

packets 15-30 as a separate substream.

3.2 Verification

The objective of this section is to evaluate the accuracy of *Pathload* with both NS simulations [64], and experiments over real Internet paths.

3.2.1 Simulation results

The following simulations evaluate the accuracy of *Pathload* in a controlled and reproducible environment under various load conditions and path configurations. Specifically, we implemented the *Pathload* sender ($\mathcal{SND}$) and receiver ($\mathcal{RCV}$) in application-layer NS modules. The functionality of these modules is identical as in the original *Pathload*, with the exception of some features that are not required in a simulator (such as the detection of context switches).

In the following, we simulate the H-hop topology of Figure 10. The *Pathload* packets enter the path in hop 1 and exit at hop H. The hop in the middle of the path is the tight link, and it has capacity C_t, avail-bw A_t, and utilization u_t. We refer to the rest of the links as *non-tight*, and consider the case where they all have the same capacity C_{nt}, avail-bw A_{nt}, and utilization u_{nt}. Cross-traffic is generated at each link from ten random sources that, unless specified otherwise, generate Pareto interarrivals with α=1.9. The cross-traffic packet sizes are distributed as follows: 40% are 40B, 50% are 550B, and 10% are 1500B. The end-to-end propagation delay in the path is 50 msec, and the links are sufficiently buffered to avoid packet losses.

We next examine sensitivity of Pathload results to f and PDT threshold, the effect of

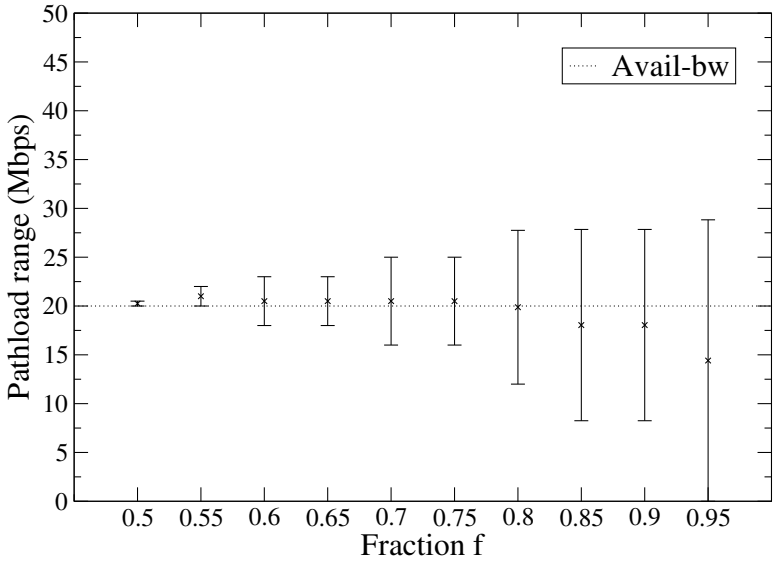


Figure 11: Simulation results for different values of fraction f.

cross-traffic load in the tight and non-tight links on the accuracy of *Pathload*, i.e., the effect of u_t and u_{nt}, as well as the effect of the number of hops H. Another important factor is the relative magnitude of the avail-bw in the non-tight links A_{nt} and in the tight link A_t. To quantify this, we define the *path tightness factor* as

$$\beta = \frac{A_{nt}}{A_t} \geq 1 \tag{20}$$

Unless specified otherwise, the default parameters in the following simulations are H=5 hops, C_t=10Mbps, β=5, u_t=u_{nt}=60%, f=0.7, PCT threshold=0.55 and PDT threshold=0.4.

Figure 11 shows the sensitivity of *Pathload* range to f. The reported *Pathload* range is a result of a single run of *Pathload*. Since C_t=50Mbps, and u_t=u_{nt}=60%, the average avail-bw in the path is A_t=20Mbps. The key thing to note in this figure is that a higher value of f increases the width of the grey region and hence the estimated avail-bw range. The reason is that, for a given R and A, a higher f implies a larger fraction of streams must be of type-I if $R > A$ (type-N if $R < A$) and hence harder to characterize entire fleet as increasing (non-increasing).

Figure 12 shows the sensitivity of *Pathload* range to PDT threshold for a single run of *Pathload*. The simulation parameters are same as before and only PDT algorithm is used to detect trend in the stream. The main observation in figure 12 is that *Pathload*

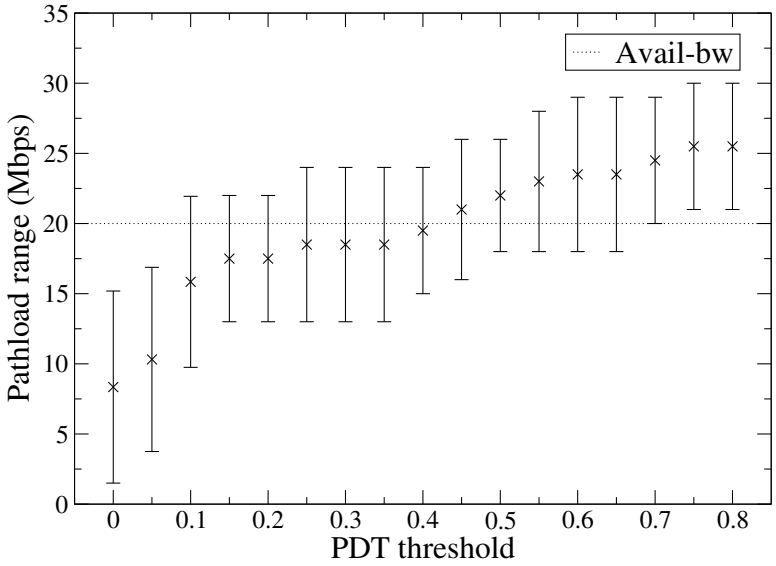


Figure 12: Simulation results for different values of PDT threshold.

underestimates the avail-bw for very low values of PDT threshold and overestimates the avail-bw for high values of PDT threshold. To understand this, note that a stream has increasing trend if $S_{pdt} >$ PDT threshold. A lower PDT threshold means that a stream can be marked of type-I even if $R < A$. Similarly, a higher PDT threshold implies that a stream can be marked as type-N even if the trend $R > A$ but the delay trend were not strictly increasing. Sensitivity analysis of PCT threshold showed similar trend as PDT threshold, however we did not to include the results due to space constraint.

Figure 13 examines the accuracy of *Pathload* in four tight link utilization values, ranging from light load (u_t=30%) to heavy load (u_t=90%). We also consider two cross-traffic models: exponential interarrivals and Pareto interarrivals with infinite variance (α=1.5). For each utilization and traffic model, we run *Pathload* 50 times to measure the avail-bw in the path. After each run, the tool reports a range $[R^{min}, R^{max}]$ in which the avail-bw varies. The *Pathload* range that we show in Figure 13 results from averaging the 50 lower bounds R^{min} and the 50 upper bounds R^{max}. The coefficient of variation for the 50 samples of R^{min} and R^{max} in the following simulations was typically between 0.10 and 0.30.

The main observation in Figure 13 is that Pathload *produces a range that includes the average avail-bw in the path, in both light and heavy load conditions at the tight link*. This

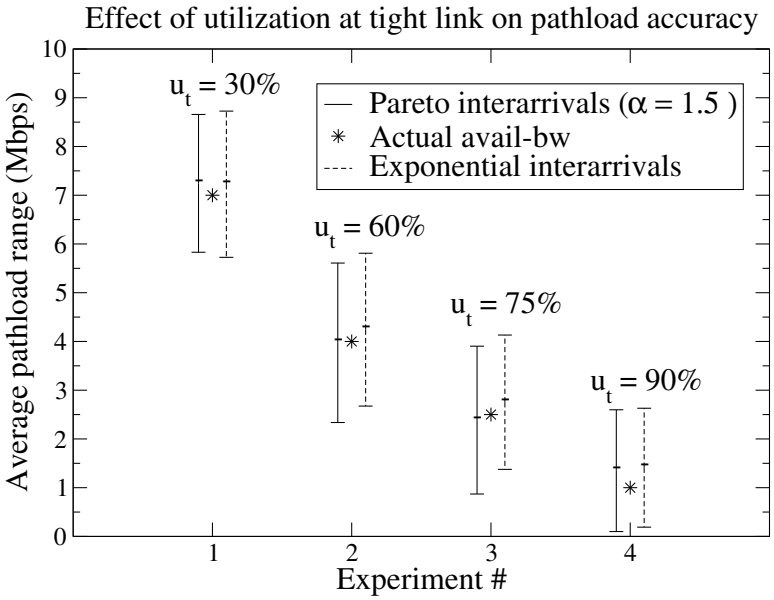


Figure 13: Simulation results for different traffic types and tight link loads.

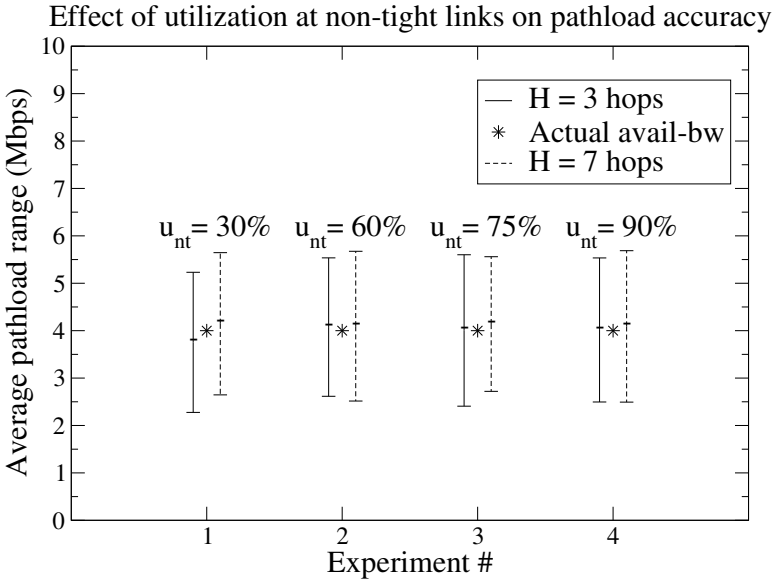


Figure 14: Simulation results for different non-tight link loads.

is true with both the smooth interarrivals of Poisson traffic, and with the infinite-variance Pareto model. For instance, when the avail-bw is 4Mbps, the average *Pathload* range in the case of Pareto interarrivals is from 2.4 to 5.6 Mbps. It is also important to note that *the center of the* Pathload *range is relatively close to the average avail-bw.* In Figure 13, the

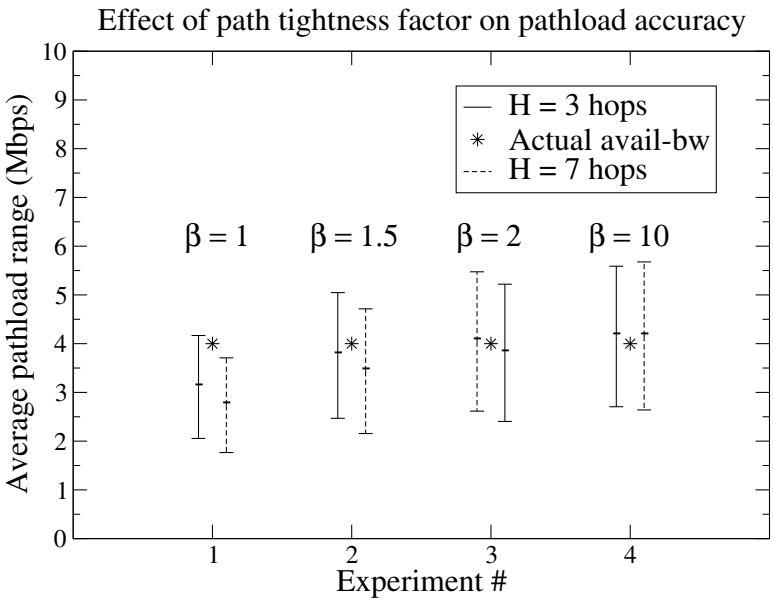


Figure 15: Simulation results for different path tightness factors β.

maximum deviation between the average avail-bw and the center of the *Pathload* range is when the former is 1Mbps and the latter is 1.5Mbps.

The next issue is whether the accuracy of *Pathload* depends on the number and load of the non-tight links. Figure 14 shows, as in the previous paragraph, 50-sample average *Pathload* ranges for four different utilization points u_{nt} at the non-tight links, and for two different path lengths H. Since C_t=10Mbps and u_t=60%, the end-to-end avail-bw in these simulations is 4Mbps. The path tightness factor is β=5, and so the avail-bw in the non-tight links is A_{nt}=20Mbps. So, even when there is significant load and queueing at the non-tight links (which is the case when u_{nt}=90%), the end-to-end avail-bw is quite lower than the avail-bw in the $H-1$ non-tight links.

The main observation in Figure 14 is that Pathload *estimates a range that includes the actual avail-bw in all cases, independent of the number of non-tight links or of their load*. Also, *the center of the* Pathload *range is within 10% of the average avail-bw* A_t. So, when the end-to-end avail-bw is mostly limited by a single link, *Pathload* is able to estimate accurately the avail-bw in a multi-hop path even when there are several other queueing points. The non-tight links introduce noise in the OWDs of *Pathload* streams, but they do

not affect the OWD trend that is formed when the stream goes through the tight link.

Finally, we examine whether the accuracy of *Pathload* depends on the path tightness factor β. Figure 15 shows 50-sample average *Pathload* ranges for four different values of β, and for two different path lengths H. As previously, C_t=10Mbps, u_t=60%, and so the average avail-bw is A_t=4 Mbps. Note that when the path tightness factor is β=1, all H links have the same avail-bw A_{nt}=A_t=4Mbps, meaning that they are all tight links. The main observation in Figure 15 is that Pathload *succeeds in estimating a range that includes the actual avail-bw when there is only one tight link in the path, but it underestimates the avail-bw when there are multiple tight links.*

To understand the nature of this problem, note that an underestimation occurs when R^{max} is set to a fleet rate R, even though R is less than the avail-bw A_t. Recall that *Pathload* sets the state variable R^{max} to a rate R when more than f=70% of a fleet's streams have an increasing delay trend. A stream of rate R, however, can get an increasing delay trend at any link of the path in which the avail-bw *during the stream* is less than R. Additionally, after a stream gets an increasing delay trend it is unlikely that it will loose that trend later in the path. Consider a path with H_t tight links, all of them having an *average* avail-bw A_t. Suppose that p is the probability that a stream of rate R will get an increasing delay trend at a tight link, even though $R < A_t$. Assuming that the avail-bw variations at different links are independent, the probability that the stream will have an increasing delay trend after H_t tight links is $1-(1-p)^{H_t}$, that increases very quickly with H_t. This explains why the underestimation error in Figure 15 appears when β is close to one (i.e., $H_t > 1$), and why it is more significant with H_t=7 rather than with 3 hops.

3.2.2 Experimental results

We have also verified *Pathload* experimentally, comparing its output with the avail-bw shown in the MRTG [65] graph of the path's tight link. MRTG is a widely used tool that displays the utilized bandwidth of a link, based on information that comes directly from the router interface [65]. Specifically, MRTG periodically reads the number of bytes sent and received from the router's Management Information Base using SNMP. The default measurement

Table 1: Measurement hosts and their locations.

Host	Location
strauss	Univ.Delaware, Newark DE, USA
wayback	Univ.Oregon, Eugene Or, USA
agwnia	Univ.Crete, Heraclion, Greece
vergina	Aristotle Univ. (AUTH), Thessaloniki, Greece
scylla	Univ.Ioannina, Ioannina, Greece

period is 5 minutes, and thus the MRTG bandwidth readings should be interpreted as 5-min averages. If the capacity of the link is also known, the avail-bw in the same time interval is the capacity minus the utilized bandwidth.

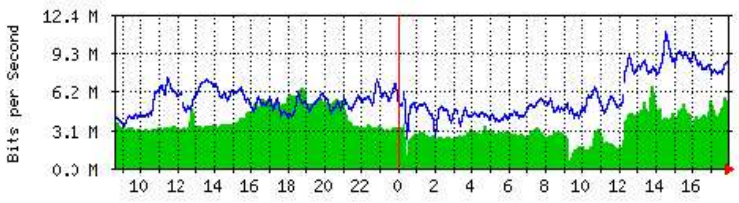


Figure 16: Example of an MRTG graph for a 14Mbps link. The x-axis is measured in hours.

Figure 16 shows a snapshot of an MRTG graph for a 14Mbps duplex link. The two curves (shaded vs line) refer to the two directions of the link (IN vs OUT).

MRTG offers a primitive way to verify *Pathload* measurements, based on 5-minute avail-bw measurements for individual links. In the paths that we experiment with, we have access to MRTG graphs for most links in the path (especially for the most heavily utilized links), as well as information about their capacity. The tight link remains the same for the duration of many hours in these paths, and so the measurement of end-to-end avail-bw is based on a single MRTG graph. Even though this verification methodology is not too accurate, it was the only way in which we could evaluate *Pathload* in real and wide-area Internet paths.

An MRTG reading is a 5-minute average avail-bw. *Pathload*, however, takes about 10-30 seconds to produce an estimate. To compare these short-term *Pathload* estimates with the MRTG average, we run *Pathload* consecutively for 5 minutes. Suppose that in a 5-min

(300sec) interval we run *Pathload* W times, and that run i lasted for q_i seconds, reporting an avail-bw range $[R_i^{min}, R_i^{max}]$ $(i = 1, \ldots W)$. The 5-min average avail-bw $\hat{R}$ that we report here for *Pathload* is the following weighted average of $(R_i^{min} + R_i^{max})/2$:

$$\hat{R} = \sum_{i=1}^{W} \frac{q_i}{300} \frac{R_i^{min} + R_i^{max}}{2} \tag{21}$$

The names and locations of the five hosts that we used in these experiments are given in Table 1. Three hosts are located at Greek universities, connected through the academic & research network GRNET [22]. The U-Delaware host (*strauss*) and the U-Oregon host (*wayback*) are connected through the Abilene network, while the Abilene is connected to GRNET through the Dante (now GEANT) network in Europe. The underlying routes in these paths are almost always the same. In these experiments, the resolution parameters were ω=1Mbps and χ=1.5Mbps, and f, PCT, and PDT thresholds were 0.7, 0.6 and 0.5, respectively.

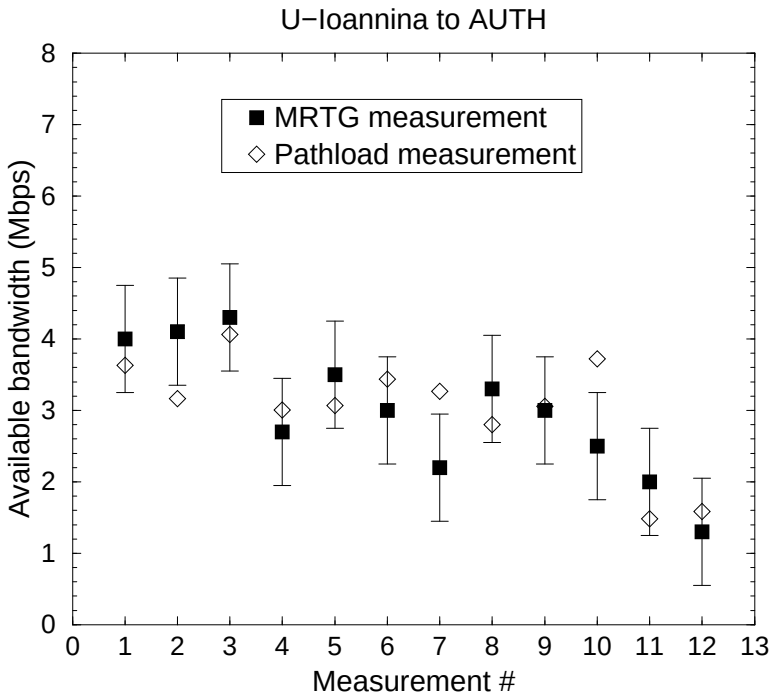


Figure 17: Verification experiment in Path #1

Figure 17 compares the MRTG and *Pathload* measurements for twelve 5-min intervals at the path from U-Ioannina to AUTH in Greece. The tight (and narrow) link in this path

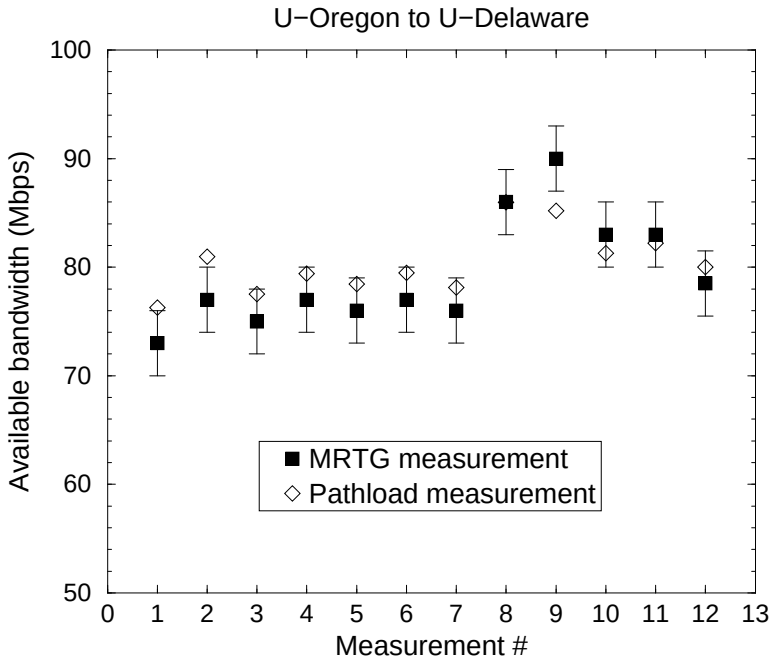


Figure 18: Verification experiment in Path #2

is the access link that connects Univ-Ioannina to GRNET (C=8.2 Mbps). The *Pathload* measurements are within the MRTG range in 9 out of 12 runs. The errors in the remaining 3 runs are less than 0.5Mbps.

Figure 18 refers to the path from U-Oregon to U-Delaware. An interesting point about this path is that the tight link is different than the narrow link. The former is an IP-over-SONET 155Mbps OC3 link that connects the Oregon GigaPOP to the Abilene, while the latter is a Fast Ethernet interface (100Mbps). The best resolution we could get from the MRTG readings in this case is 6Mbps. The *Pathload* measurements are within the MRTG range in 10 out of 12 runs, with an error of less than 2Mbps in the remaining runs.

Figure 19 refers to the path from U-Delaware to U-Crete. The tight & narrow link is the access link that connects U-Crete to GRNET (C=12.4Mbps). The *Pathload* measurements are within the MRTG range in 9 out of 12 cases, with marginal errors in the remaining runs.

Figure 20 refers to the path from U-Ioannina and U-Crete. The tight & narrow link is the U-Ioannina access link. The *Pathload* measurements are within the MRTG ranges in

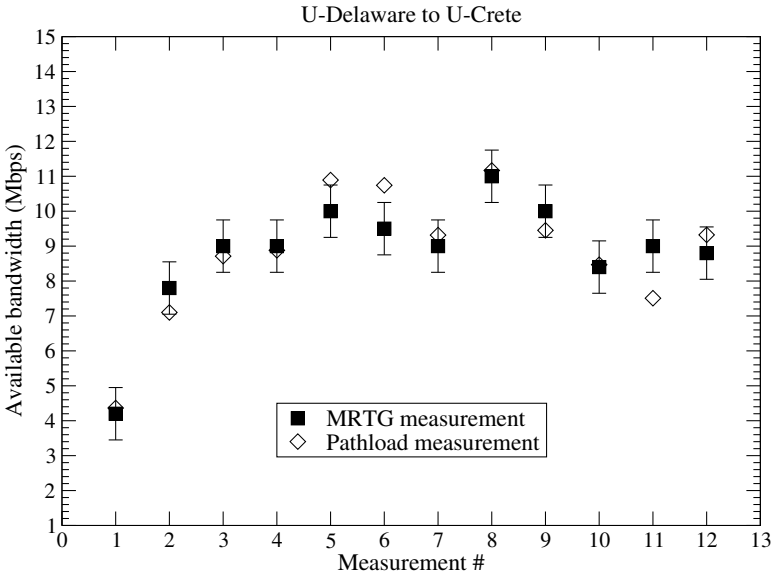


Figure 19: Verification experiment in Path #3

11 out of 12 runs.

Finally, Figure 21 refers to the path from U-Ioannina to U-Oregon. Again, the tight link is the U-Ioannina access link. The *Pathload* measurements are within the MRTG ranges in 8 out of 12 runs, with an an error of less than 0.5Mbps in the remaining runs.

More recently, Pathload has also been evaluated by independent researchers and it has been shown to be more accurate compared to other tools [80, 23]. Figure 22 illustrates a result from a study done Shriram et al. comparing the performance of Pathload with more recent avail-bw estimation tools.

3.3 Available bandwidth dynamics

In this section, we use *Pathload* to evaluate the variability of the avail-bw in different timescales and operating conditions. Given that our experiments are limited to a few paths, we do not attempt to make quantitative statements about the avail-bw variability in the Internet. Instead, our objective is to show the *relative effect* of certain operational factors on the variability of the avail-bw.

In the following experiments, ω=1Mbps and χ=1.5Mbps. Note that because $\omega < \chi$, *Pathload* terminates due to the ω constraint only if there is no grey-region; otherwise, it

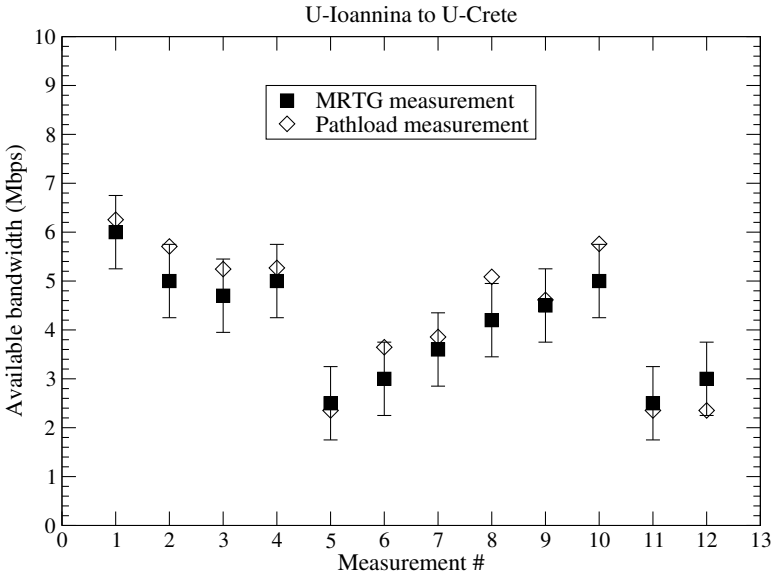


Figure 20: Verification experiment in Path #4

exits due to the χ constraint. So, the final range $[R^{min}, R^{max}]$ that *Pathload* reports is either at most 1Mbps (ω) wide, indicating that there is no grey-region, or it overestimates the width of the grey-region by at most 2χ. Thus, $[R^{min}, R^{max}]$ *is within* 2χ *(or* ω*) of the range in which the avail-bw varied during that* Pathload *run.* To compare the variability of the avail-bw across different operating conditions and paths, we define the following *relative variation* metric:

$$\rho = \frac{R^{max} - R^{min}}{(R^{max} + R^{min})/2} \tag{22}$$

In the following graphs, we plot the {5,15, ... 95} percentiles of ρ based on 110 *Pathload* runs for each experiment.

3.3.1 Variability and load conditions

Figure 23 shows the CDF of ρ for a path with C=12.4Mbps in three different utilization ranges of the tight link: u=20-30%, 40-50%, and 75-85%. Notice that *the variability of the avail-bw increases significantly as the utilization* u *of the tight link increases (i.e., as the avail-bw* A *decreases).* This observation is not a surprise. In Markovian queues, say in $M|M|1$, the variance of the queueing delay is inversely proportional to the square of the avail-bw. The fact that increasing load causes higher variability was also observed

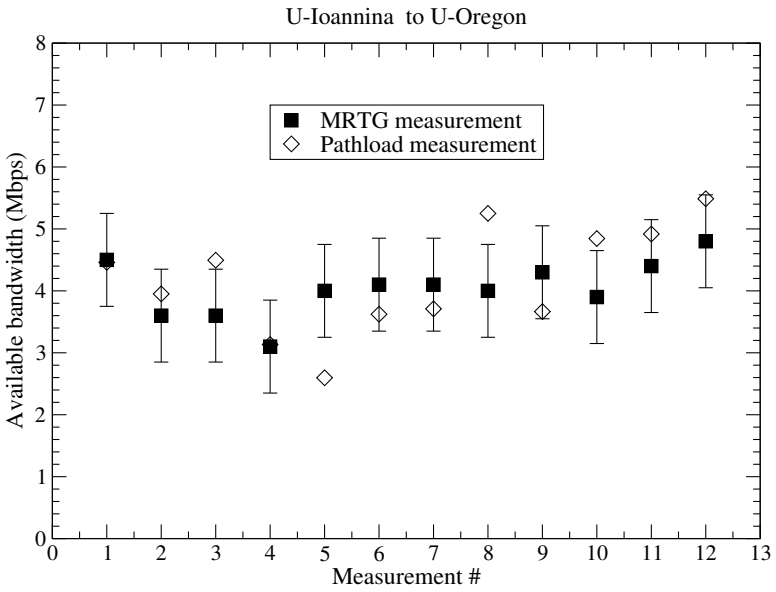


Figure 21: Verification experiment in Path #5

for self-similar traffic in [53]. Returning to Figure 23, the 75-percentile shows that when the avail-bw is around A=9Mbps (u=20-30%), three quarters of the measurements have a relative variation $\rho \leq 0.25$. In heavy-load conditions (u=75-85%) on the other hand, when A=2-3Mbps, the same fraction of measurements give almost five times higher relative variation ($\rho \leq 1.3$).

We observed a similar trend in all the paths that we experimented with. For users, this suggests that a lightly loaded network will not only provide more avail-bw, but also a more predictable and smooth throughput. This latter attribute is even more important for some applications, such as streaming audio/video.

3.3.2 Variability and statistical multiplexing

In this experiment, we run *Pathload* in three different paths, (A), (B), and (C), when the tight link utilization was roughly the same (around 65%) in all paths. The capacity of the tight link is 155Mbps (A), 12.4Mbps (B), and 6.1Mbps (C). The tight link in (A) connects the Oregon GigaPoP to the Abilene network, the tight link in (B) connects a large university in Greece (Univ-Crete) to a national network (GRnet), while the tight link in (C) connects a smaller university (Univ-Pireaus) to the same national network. Based

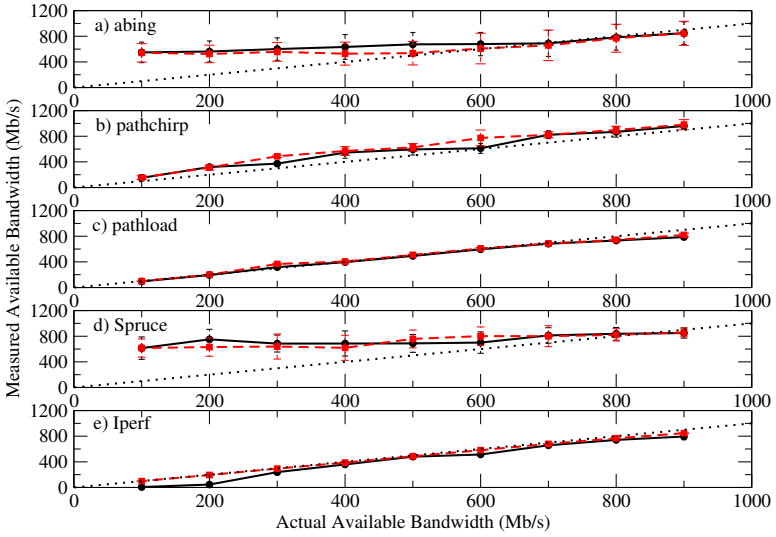


Figure 22: Pathload evaluation result from Shriram et al. [80]

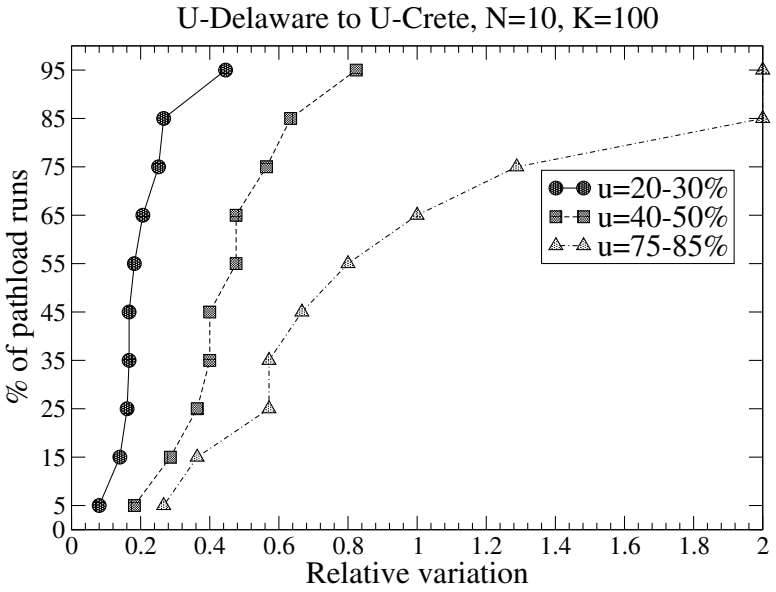


Figure 23: Variability of avail-bw in different load conditions.

on these differences, it is reasonable to *assume* that the degree of statistical multiplexing, *i.e., the number of flows that simultaneously use the tight link*, is highest in path (A), and higher in (B) than in (C). Figure 24 shows the CDF of ρ in each path. If our assumption about the number of simultaneous flows in the tight link of these paths is correct, we observe that *the variability of the avail-bw decreases significantly as the degree of statistical*

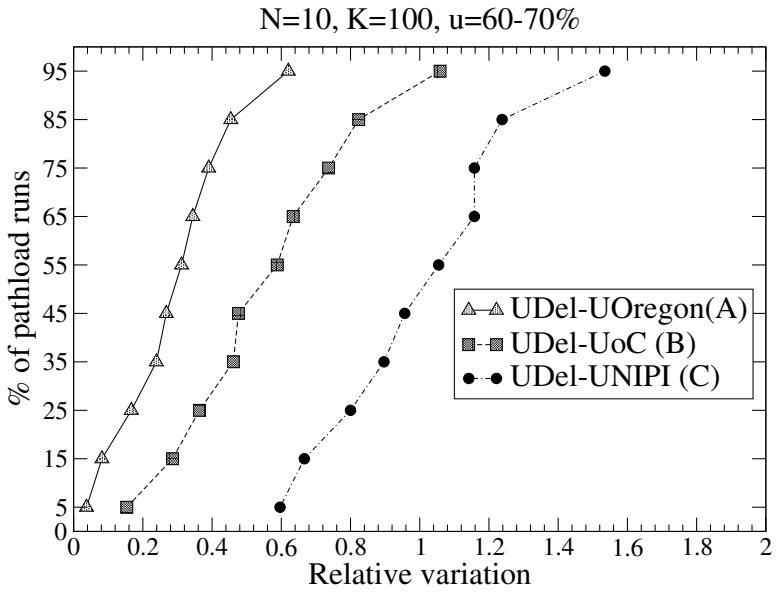


Figure 24: Variability of avail-bw in different paths.

multiplexing increases. Specifically, looking at the 75-percentile, the relative variation is $\rho \leq 0.4$ in path (A), it increases by almost a factor of two ($\rho \leq 0.74$) in path (B), and by almost a factor of three ($\rho \leq 1.18$) in path (C). It should be noted, however, that there may be other differences between the three paths that cause the observed variability differences; the degree of statistical multiplexing is simply one plausible explanation.

For users, the previous measurements suggest that if they can choose between two networks that operate at about the same utilization, the network with the wider pipes, and thus with a higher degree of statistical multiplexing, will offer them a more predictable throughput. For network providers, on the other hand, it is better to aggregate traffic in a higher-capacity trunk than to split traffic in multiple parallel links of lower capacity, if they want to reduce the avail-bw variability.

3.3.3 The effect of the stream length

Since $V=KT$, the stream duration V increases proportionally to the stream length K. With a longer stream, we examine whether $R > A$ over wider timescales. As mentioned in the introduction, however, the variability of the avail-bw A decreases as the averaging timescale increases. So, the variability in the relation between R and A, and thus the variability of

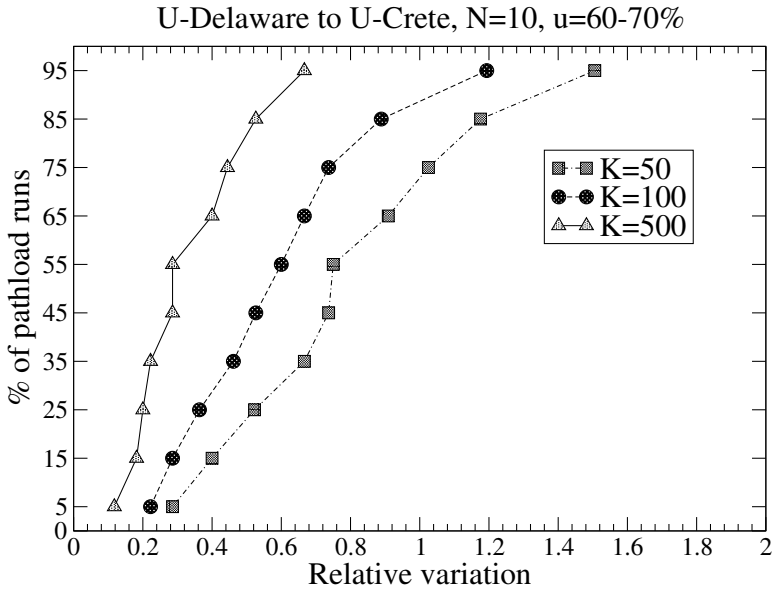


Figure 25: The effect of K on the variability of the avail-bw.

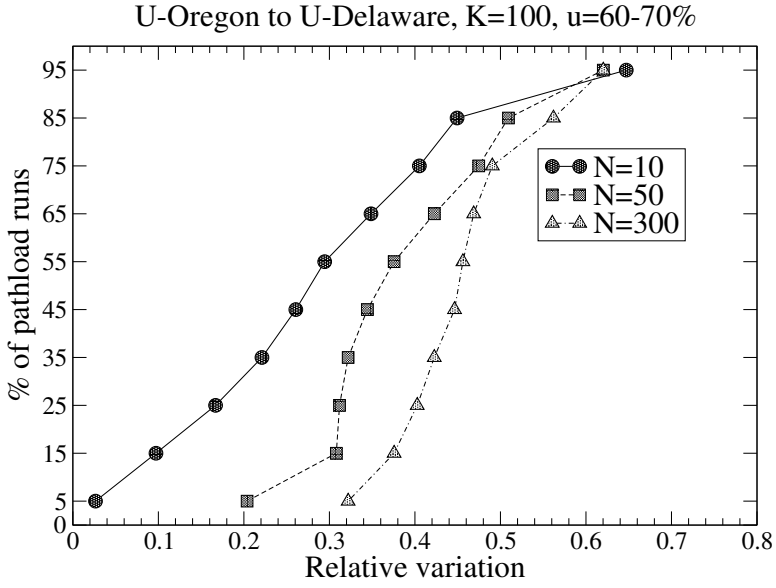


Figure 26: The effect of N on the variability of the avail-bw.

the *Pathload* measurements, is expected to decrease as the stream duration increases.

Figure 25 shows the CDF of ρ for three different values of K in a path with C=12.4Mbps. During the measurements, A was approximately 4.5Mbps. The stream duration V for $R = A$, L=200B, and T=356μs is 18ms for K=50, 36ms for K=100, and 180ms for K=500. The major observation here is that *the variability of the avail-bw decreases significantly as the stream duration increases*, as expected. Specifically, when V=180ms, 75% of the measurements produced a range that is less than 2.0Mbps wide ($\rho \leq$0.45). When V=18ms, on the other hand, the corresponding maximum avail-bw range increases to 4.7Mbps ($\rho \leq$1.05).

3.3.4 The effect of the fleet length

Suppose that we measure the avail-bw $A^\tau(t)$ in a time interval $(t_0, t_0 + \Theta)$, with a certain averaging timescale τ ($\tau < \Theta$). These measurements will produce a range of avail-bw values, say from a minimum A^τ_{min} to a maximum $A^\tau_{\max}$. If we keep τ fixed and increase the measurement period Θ, the range $[A^\tau_{min}, A^\tau_{\max}]$ becomes wider because it tracks the boundaries of the avail-bw process during a longer time period. An additional effect is that, as Θ increases, the variation of the width $A^\tau_{\max} - A^\tau_{min}$ decreases. The reason is that the boundaries A^τ_{min} and $A^\tau_{\max}$ tend to their expected values (assuming a stationary process), as the duration of the measurement increases.

The measurement period Θ is related to the number of streams N in a fleet, and to the fleet duration $U = N(V + \Delta)$. As we increase N, keeping V fixed, we expand the time window in which we examine the relation between R and A, and thus we increase the likelihood that the rate R will be in the grey-region of the avail-bw ($R \bowtie A$). So, the grey-region at the end of the *Pathload* run will be wider, causing a larger relative variation ρ. This effect is shown in Figure 26 for three values of N. Observe that *as the fleet duration increases, the variability in the measured avail-bw increases.* Also, *as the fleet duration increases, the variation across different* Pathload *runs decreases*, causing a steeper CDF.

3.4 TCP and available bandwidth

We next focus on the relationship between the avail-bw in a path and the throughput of a persistent (or 'bulk') TCP connection with arbitrarily large advertised window. There are

two questions that we attempt to explore. First, can a bulk TCP connection measure the avail-bw in a path, and how *accurate* is such an avail-bw measurement approach? Second, what happens then to the rest of the traffic in the path, i.e., how *intrusive* is a TCP-based avail-bw measurement?

It is well-known that the throughput of a TCP connection can be limited by a number of factors, including the receiver's advertised window, total transfer size, RTT, buffer size at the path routers, probability of random losses, and avail-bw in the forward and reverse paths. In the following, we use the term *Bulk Transfer Capacity (BTC) connection* (in relation to [58]) to indicate a TCP connection that is only limited by the network, and not by end-host constraints.

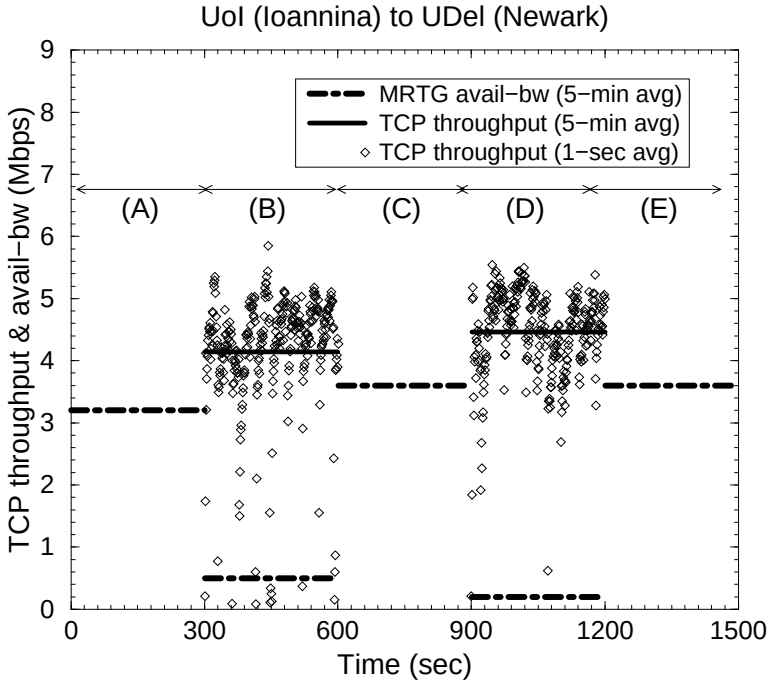


Figure 27: Available bandwidth and BTC throughput.

Let us first describe the results of an experiment that measures the throughput of a BTC connection from Univ-Ioannina (Greece) to Univ-Delaware. The TCP sender was a SunOS 5.7 box, while the TCP receiver run FreeBSD 4.3. The tight link in the path has a capacity of 8.2Mbps. Consider a 25-minute (min) measurement interval, partitioned in five consecutive 5-min intervals (A), (B), (C), (D), and (E). During (B) and (D) we perform a

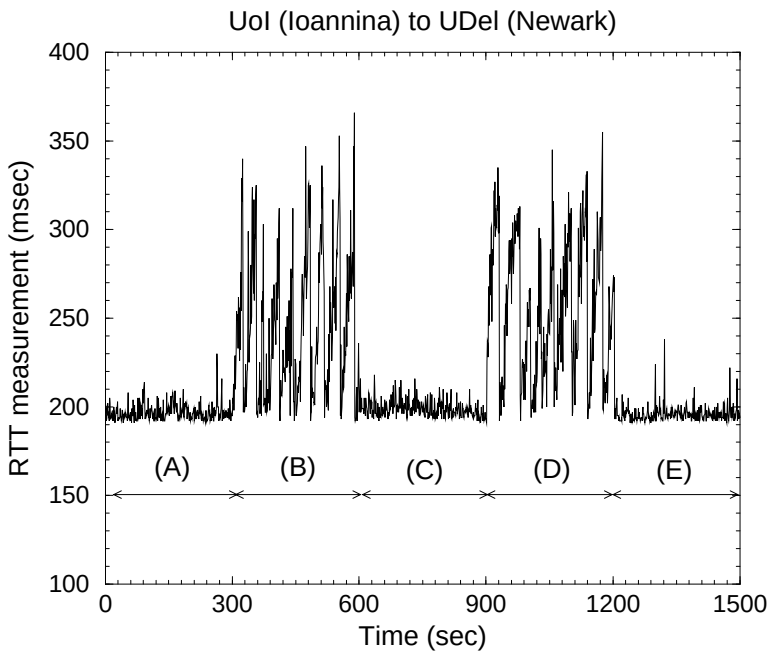


Figure 28: RTT measurements during the experiment of Figure 27.

BTC connection, and measure its throughput in both 1-sec intervals and in the entire 5-min interval. During (A), (C), and (E), we do not perform a BTC connection. Throughout the 25-min interval, we also monitor the avail-bw in the path using MRTG data for each of the 5-min intervals. In parallel, we use *ping* to measure the RTT in the path at every second. Figure 27 shows the throughput of the two BTC connections, as well as the 5-min average avail-bw in the path, while Figure 28 shows the corresponding RTT measurements.

We make three important observations from these figures. First, the avail-bw during (B) and (D) is less than 0.5Mbps, and so *for most practical purposes, the BTC connection manages to saturate the path*. Also note, however, that the BTC throughput shows high variability when measured in 1-sec intervals, often being as low as a few hundreds of Kbps. Consequently, even though a bulk TCP connection that lasts for several minutes should be able to saturate a path when not limited by the end-hosts[2], *shorter TCP connections should expect a significant variability in their throughput*.

Second, *there is a significant increase in the RTT measurements during (B) and (D),*

[2]This will not be the case however in under-buffered links, or paths with random losses [50].

from a 'quiescent point' of 200ms to a high variability range between 200ms and 370ms. The increased RTT measurements can be explained as follows: the BTC connection increases its congestion window until a loss occurs. A loss however does not occur until the queue of the tight link overflows[3]. Thus, the queue size at the tight link increases significantly during the BTC connection, causing the large RTT measurements shown. To quantify the queue size increase, note that the maximum RTTs climb up to 370ms, or 170ms more than their quiescent point. The tight link has a capacity of 8.2Mbps, and so its queue size becomes occasionally at least 170KB larger during the BTC connection. *The RTT jitter is also significantly higher during (B) and (D)*, as the queue size of the tight link varies between high and low occupancy due to the 'sawtooth' behavior of the BTC congestion window.

Third, the BTC connection gets an average throughput during (B) and (D), that is about 20-30% more than the avail-bw in the surrounding intervals (A), (C), and (E). This indicates that *a BTC connection can get more bandwidth than what was previously available in the path, grabbing part of the throughput of other TCP connections.* To see how this happens, note that the presence of the BTC connection during (B) and (D) increases the RTT of all other TCP connections that go through the tight link, because of a longer queue at that link. Additionally, the BTC connection causes buffer overflows, and thus potential losses to other TCP flows[4]. The increased RTTs and losses reduce the throughput of other TCP flows, allowing the BTC connection to get a larger share of the tight link than what was previously available.

To summarize, a BTC connection measures more than the avail-bw in the path, because it shares some of the previously utilized bandwidth of other TCP connections, and it causes significant increases in the delays and jitter at the tight link of its path. This latter issue is crucial for real-time and streaming applications that may be active during the BTC connection.

[3]Assuming Drop-Tail queueing, which is the common practice today.
[4]We did not observe however losses of *ping* packets.

3.5 Is Pathload intrusive?

An important question is whether *Pathload* has an *intrusive behavior*. There are several ways to define intrusiveness for a avail-bw estimation tool. For instance, the tool can be characterized as intrusive depending on whether it injects probing traffic into the network i.e., based on active or passive nature of estimation methodology, or based on whether the peak probing rate is greater than a certain threshold. We define intrusiveness based on whether the probing causes a significant decrease in the avail-bw, and increased delays or losses. The intuition behind this definition is that a slight increase in average delay or loss rate will not have any significant impact on the data traffic in a network path.

Figures 29 and 30 show the results of a 25-min experiment, performed similarly with the experiment of §3.4. Specifically, *Pathload* runs during the 5-min intervals (B) and (D). We monitor the 5-min average avail-bw in the path using MRTG (Figure 29), and also perform RTT measurements in every 100ms (Figure 30). The RTT measurement period here is ten times smaller than in §3.4, because we want to examine whether *Pathload* causes increased delays or losses even in smaller timescales than one second.

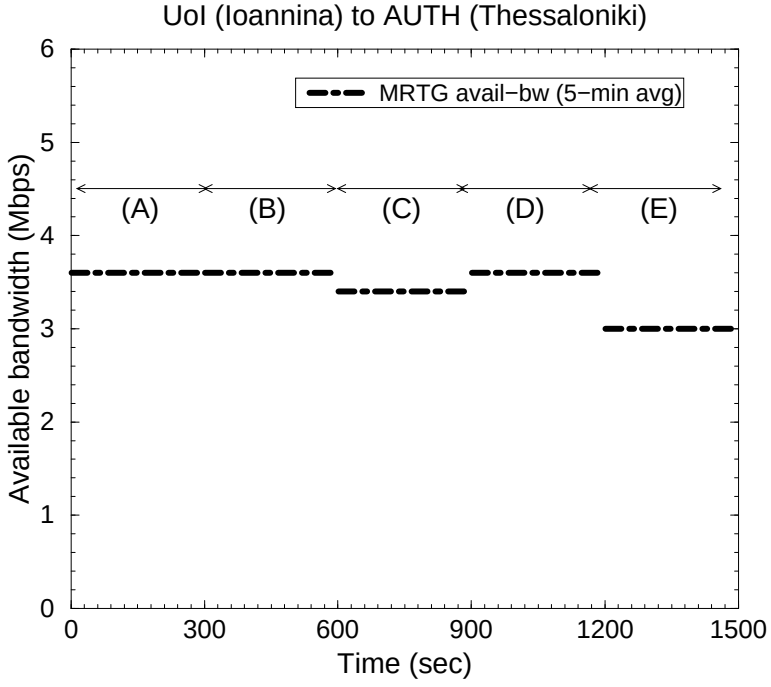


Figure 29: Available bandwidth measurements.

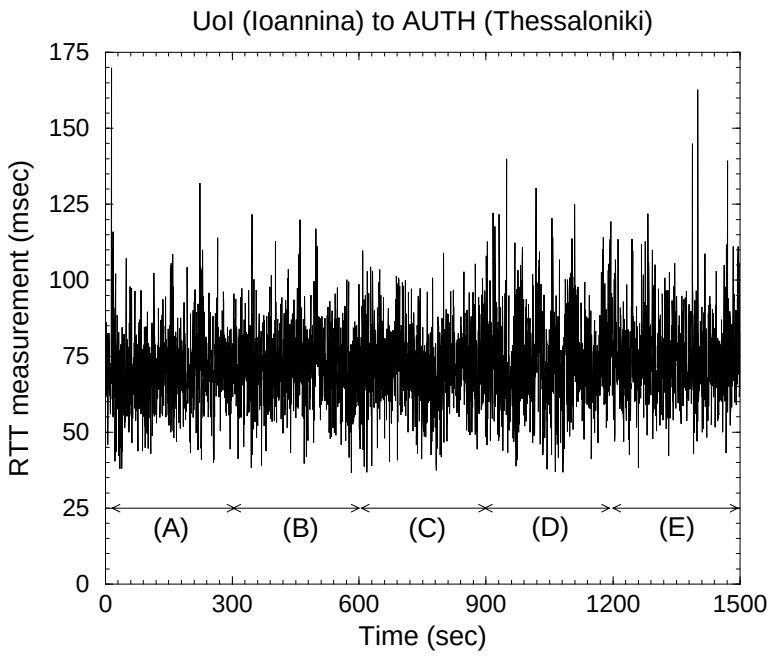


Figure 30: RTT measurements during the experiment of Figure 29.

The results of the experiment are summarized as follows. First, the avail-bw during (B) and (D) does not show a measurable decrease compared to the intervals (A), (C), and (E). Second, the RTT measurements do not show any measurable increase when *Pathload* runs. So, *Pathload* does not seem to cause a persistent queue size increase, despite the fact that it often sends streams of higher rate than the avail-bw. The reason is that each stream is only K=100 packets, and a stream is never sent before the previous one has been acknowledged. We also note that none of the *Pathload* streams encountered any losses in this experiment. None of the *ping* packets was lost either.

Next, we examine the overhead and the measurement latency of Pathload using the simulation setup described in §3.2.1. Specifically, we simulate the 3-hop topology of Figure 10. The hop in the middle of the path is the tight link, with a capacity of 10 Mbps. The non-tight link capacity is 100 Mbps and the end-to-end propagation delay in the path is 50 msec. Cross traffic is generated at each link from 5 exponential sources. For each utilization, we run Pathload 10 times to measure the avail-bw in the path.

Figure 31 shows the average measurement latency of Pathload with respect to the average utilization. As discussed in §3.1.6, it is difficult to predict the latency of a Pathload run due to iterative nature of the algorithm. We, however, know that the measurement latency of Pathload depends on both the fleet duration U and the number of fleets F, and it can be expressed as $U \times F$. As described in §3.1.4, the fleet duration is given as $U = N(KT + \Delta)$, which is independent of the average utilization (or avail-bw). The value of F, however, depends on the average avail-bw and generally increases with the average avail-bw. The increase in value of F explains the observed increase in measurement latency of Pathload.

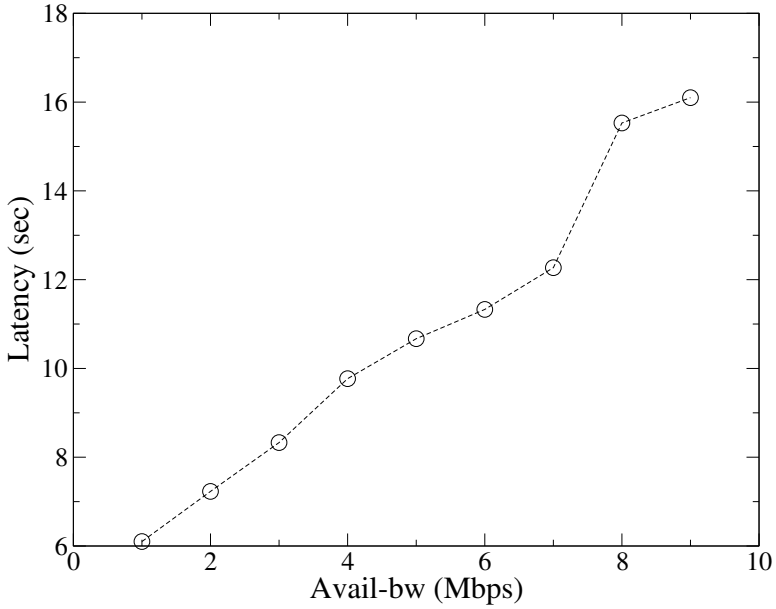


Figure 31: Measurement latency in a 100 msec round-trip time path.

Figure 32 shows the overhead of Pathload, measured in terms of the average sending rate, under different average utilization at the tight link. Note that the average sending rate is always less than the average avail-bw in absolute terms. In relative terms, the average sending rate is less than 25% of the average avail-bw in the majority of cases.

3.6 Summary

In this chapter, we showed the implementation of SLoPS in a tool called *Pathload*. We showed through simulations that *Pathload* accurately measures avail-bw under varying load conditions and path configurations. Additionally, we verified the tool experimentally on

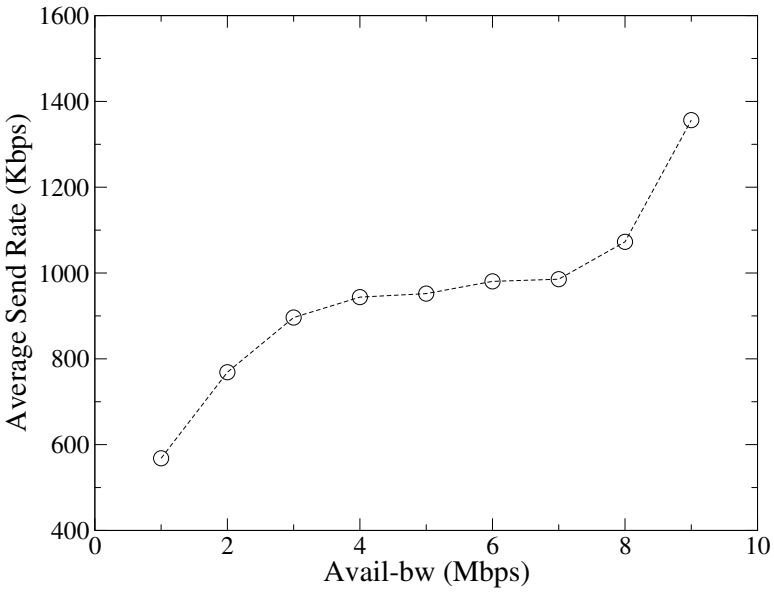


Figure 32: Average sending rate in a 100 msec round-trip time path.

several Internet paths and also show that Pathload is non-intrusive. We also examined the variability of avail-bw using *Pathload* in different paths and load conditions. Finally, we also examined the relationship between persistent TCP throughput and avail-bw. In the next chapter, we present algorithms and a tool to estimate variability of avail-bw.

CHAPTER IV

AVAILABLE BANDWIDTH VARIATION RANGE ESTIMATION: PATHVAR

4.1 Variation Range of Avail-bw

Several estimation techniques and tools based on active measurements have been developed, including Delphi [77], TOPP [59], Pathload [33], IGI/PTR [25], Pathchirp [78], Spruce [81], and Bfind [3], for the estimation of avail-bw. All previous work aimed to estimate the *average avail-bw*, largely ignoring that the avail-bw is a time-varying quantity, defined as an average over a certain *measurement time scale*. If we view the avail-bw as a stationary random process, the second-order statistics, namely the variance of the marginal distribution and the autocorrelation function, are needed for a more complete characterization of the avail-bw process. In this work, we focus on the end-to-end estimation of the variability of the avail-bw marginal distribution, leaving the identification of the correlation structure for future work.

The avail-bw, especially in sub-second scales, can exhibit significant variations around its time average, making the latter a rather poor-quality estimator or predictor. To illustrate this point, Figure 33 shows the avail-bw time series, and the corresponding marginal distribution, for two measurement time scales: 20msec and 1sec. This time series was obtained from a packet trace collected at an OC-3 link, and it thus represents an exact (rather than estimated) sample path of the avail-bw process in that link. Notice that the 10%-90% variation range of the distribution in the 20msec scale is approximately 30Mbps to 75Mbps, while the average avail-bw is 52Mbps. We anticipate that information for the variation range of the avail-bw distribution will actually be more important for some applications than an estimate of the mean. For example, a video streaming application with a nominal transmission rate of 3Mbps may prefer to use a path with average avail-bw 5Mbps and a

very narrow variation range, rather than a path with average avail-bw 10Mbps but a variation range of 1Mbps to 20Mbps. Also, the measurement time scale is an application-specific parameter, and it represents the minimum time interval in which the avail-bw variations matter for a particular application.

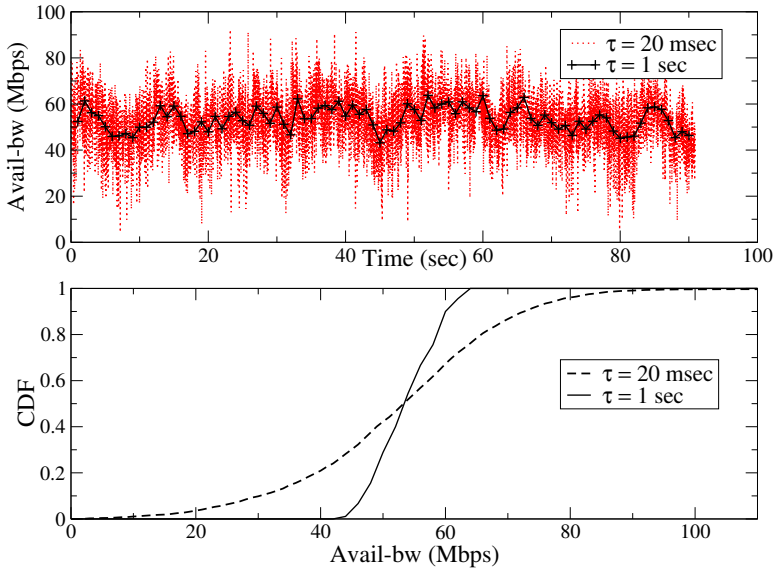


Figure 33: Top: Time series of the avail-bw process at an OC-3 link in two measurement time scales. Bottom: Empirical CDFs of the two time series.

4.1.1 Definitions

We first define the avail-bw at a network link and then at an end-to-end path. Suppose that a network path consists of H store-and-forward first-come first-served links. A link i has an instantaneous utilization $u_i(t)$ at time t; $u_i(t)$=0 if the link is idle and $u_i(t)$=1 if the link transmits a packet at time t. The average utilization $u_i(t, t+\tau)$ of link i during the time interval $(t, t+\tau)$ is

$$u_i(t, t+\tau) = \frac{1}{\tau} \int_t^{t+\tau} u_i(t)\, dt \tag{23}$$

We refer to τ as the *measurement time scale*.

The available bandwidth $A_i(t, t+\tau)$ of link i during the time interval $(t, t+\tau)$ is defined as the average residual capacity in that interval,

$$A_i(t, t+\tau) = C_i[1 - u_i(t, t+\tau)] \tag{24}$$

Consider now a network path that traverses a sequence of H links. The end-to-end available bandwidth $A(t, t+\tau)$ of the network path during $(t, t+\tau)$ is defined as the minimum avail-bw among the H links in the same interval,

$$A(t, t+\tau) = \min_{i=1...H} A_i(t, t+\tau) \tag{25}$$

We refer to the link with the minimum avail-bw as *tight link* and denote its capacity by C_t. The link with the minimum capacity is referred to as *narrow link* and has a capacity C_n. Note that, in general, the narrow and tight links can be different. Also, all existing avail-bw estimation techniques assume that the tight link has much lower avail-bw than the other links. Otherwise, the path avail-bw may be limited by more than one links. Furthermore, if there are multiple bottlenecks in a path then all existing avail-bw estimation techniques suffer from underestimation errors, which is discussed in chapter 7. In this chapter, we adopt the assumption that the path has a clearly distinguishable tight link, meaning that the avail-bw in all other links is significantly larger.

The average end-to-end avail-bw $A(t, t+\tau)$ is a function of time t and therefore it can be viewed as a random process $A_\tau(t)$, where τ is the measurement time scale. If we assume that this process is stationary and identically distributed along the time axis, then at any time instant t the process is described by the same random variable A_τ. Let $F_\tau(a)$ be the cumulative distribution function of A_τ, where $F_\tau(a) = P(A_\tau \leq a)$. The p-th percentile of the avail-bw random variable A_τ, with $p \in (0, 1)$, is the value A_τ^p such that $F_\tau(A_\tau^p) = p$; in the rest of the chapter we assume that A_τ^p is unique.

Our main objective is to estimate the variability of A_τ. One possibility could be to estimate the variance of A_τ. That would be the obvious variability metric if we knew that the avail-bw distribution is symmetric around the mean and close to Gaussian. A more general metric, however, is a percentile-based definition of variability. Specifically, if p^L is a low probability and p^H is a high probability, then we can define the variation range of A_τ as the interval $[A_\tau^L, A_\tau^H]$, where A_τ^L and A_τ^H are the p^L and p^H percentiles of A_τ, respectively. In the rest of this chapter, unless noted otherwise, we assume that the user is interested in the 10%-90% variation range, i.e., p^L=0.1 and p^H=0.9. Of course the actual definition of

the variation range would be application-specific.

Some further discussion on the relation between the measurement time scale τ and the variability of the avail-bw process is important. The mean of A_τ does not depend on τ. The variance σ_τ^2=Var[A_τ], however, depends strongly on τ and on the correlation structure of the random process $A_\tau(t)$. In general, as τ increases, the variance σ_τ^2 decreases. The speed with which the variance decreases, however, depends on the correlation structure of the underlying process. For instance, the variance of a self-similar process decreases much more slowly with τ than the variance of an IID process [66]. We return to this point in § 4.6.3.

4.1.2 Main Contributions and Overview

In this chapter, we first present a measurement technique, referred to as *percentile sampling*, that can associate a given probing rate with a percentile of the avail-bw distribution. We then use percentile sampling to design two estimation algorithms for the avail-bw variation range.

The first algorithm is iterative in nature. We refer to it as *non-parametric*, because it does not assume a specific avail-bw distribution. The non-parametric algorithm is more appropriate for very short values of the measurement time scale (typically less than 100msec) or in bottlenecks with limited flow multiplexing, where the avail-bw distribution may be non-Gaussian.

The second algorithm is parametric, because it assumes that the avail-bw follows the Gaussian distribution. This assumption is typically valid when τ >100-200msec and when the tight link carries a significant amount of aggregated traffic [46]. The parametric algorithm can produce an estimate faster than the non-parametric algorithm because it is not iterative.

The two estimation algorithms have been implemented in a measurement tool called Pathvar. We have validated Pathvar with simulations and testbed experiments using realistic Internet traffic. Pathvar can track the actual avail-bw variation range within 10-20%, even under non-stationary conditions.

Finally, we focus on four factors that can significantly affect the variation range of the avail-bw. These factors are the traffic load, number of competing flows, rate of competing flows, as well as the measurement time scale. The results of that study explain why the avail-bw appears as less or more variable depending on the load conditions and the degree of statistical multiplexing at the tight link.

A related area in the literature is that of traffic modeling and analysis, and in particular, the measurement of the second-order statistics (variance and autocorrelation) in various time scales using packet traces. That area, which started with the seminal Bellcore work [53], revealed that the traffic count process at a network link is asymptotically self-similar. The reader is referred to [66] for a survey of the related literature. Our approach and objectives in this work are significantly different. First, instead of passive traffic measurements at a single link we are interested in the active estimation of the avail-bw in an end-to-end path. Second, instead of focusing on the scaling properties of the avail-bw process, we focus on the variability of the marginal distribution at a given time scale. Third, our high-level goal is to develop tools that can be used in practice to measure important path characteristics, rather than to statistically characterize or model network traffic.

The rest of the chapter is structured as follows. The percentile sampling technique is described in §4.2. §4.3 presents the non-parametric estimation algorithm, while §4.4 presents the parametric algorithm. The implementation of Pathvar, and a few typical validation results, are summarized in §4.5. Finally, we examine the four factors that affect the variability of the avail-bw process in §4.6.

4.2 Percentile sampling

In this section, we first describe the basic technique of percentile sampling, which forms the basis of the proposed estimation algorithms in the next two sections. A number N of probing streams of duration τ and rate R are sent to a path. Each stream provides an indication of whether the avail-bw in the corresponding time interval is higher than R. The resulting N binary samples are used to estimate the percentile of the avail-bw distribution that corresponds to rate R. We also derive the required number of samples N for a given

maximum error, assuming independent sampling.

4.2.1 Basic idea

Consider a network path. The avail-bw random process measured in time scale τ is $A_\tau(t)$. As mentioned in the Introduction, we assume that this process is stationary and identically distributed. Given the previous assumptions, we can focus on the random variable A_τ and on its time-invariant marginal distribution F_τ.

The sender transmits a probing packet stream of rate R and duration τ during $(t, t+\tau)$ to the receiver. If M is the packet size, then the interarrival between successive packets is M/R and the number of probing packets is $\lceil \frac{\tau R}{M} \rceil$. The avail-bw during $(t, t+\tau)$ is given by a realization of the random variable A_τ. The receiver classifies the stream as *type-G* if it infers that the probing rate R is *greater* (or equal) than A_τ. Otherwise, the stream is classified as *type-L* (for "lower"). The exact algorithm for the classification of a stream as type-G or type-L is described in the technical report [36]. At a high level, the algorithm is based on the statistical analysis of the one-way delays of the stream's probing packet, similar to the approach taken in [33].

We use the indicator variable $I(R)$ to represent whether a stream is of type-G ($I(R) = 1$) or type-L ($I(R) = 0$). If $F_\tau(a)$ is the Cumulative Distribution Function (CDF) of A_τ, we have that

$$I(R) = \begin{cases} 1 \text{ with probability } F_\tau(R) \\ 0 \text{ with probability } 1 - F_\tau(R) \end{cases}$$

So, the expected value of $I(R)$ is $E[I(R)] = F_\tau(R)$.

A single probing stream can only tell us if the probing rate R is greater than the realization of the avail-bw random variable in the corresponding time interval. To accumulate N such samples, the sender transmits N identical probing streams[1]. The indicator variable for each stream is denoted by $I_i(R)$. Because different streams will sample different realizations of A_τ, some streams may be classified as type-G and others as type-L. Let $I(R, N)$ be the

[1]The time period between streams should be sufficiently long for the streams to not get queued behind each other while in transit.

number of streams of type-G, i.e., $I(R,N) = \sum_{i=1}^{N} I_i(R)$. The expected value of $I(R,N)$ is

$$\begin{aligned} E[I(R,N)] &= \sum_{i=1}^{N} E[I_i(R)] \\ &= F_\tau(R)N \end{aligned} \tag{26}$$

The following proposition summarizes the basic idea of percentile sampling:

Proposition 3 *For a stationary avail-bw process, the fraction $I(R,N)/N$ of type-G probing streams of rate R is an unbiased estimator of $p = F_\tau(R)$.*

Proposition 3 provides us with a mapping from a given probing rate R to the corresponding cumulative probability in the avail-bw distribution. Since our goal is to estimate a given percentile of the avail-bw distribution, we are interested in the inverse mapping, from a certain probability to the corresponding probing rate. We present two algorithms that perform this inverse mapping in § 4.3 and § 4.4.

It is important to note that Equation (26) does not require the statistical independence of the N avail-bw samples. Therefore, Proposition 3 can be used even without information about the (generally complex and unknown) correlation structure in the process $A_\tau(t)$.

4.2.2 How large should N be?

Proposition 1 refers to the expected value of $I(R,N)$. The obvious question is how large should the sample size N be so that the fraction $I(R,N)/N$ is a good approximation of $F_\tau(R)$? In this section, we derive the minimum value of N that is required for a given error tolerance.

Suppose that we aim to estimate the p-th percentile of A_τ, denoted by A_τ^p. Let ρ be the maximum allowed percentile error in the estimation of A_τ^p. This means that probing rate R would be an acceptable estimate of A_τ^p if the corresponding fraction $I(R,N)/N$ is between $p-\rho$ and $p+\rho$. So, a rate R will be correctly mapped to the p-percentile as long as

$$Prob[N(p-\rho) \leq I(R,N) \leq N(p+\rho)] > 1-\epsilon \tag{27}$$

where ϵ is a small mis-classification probability.

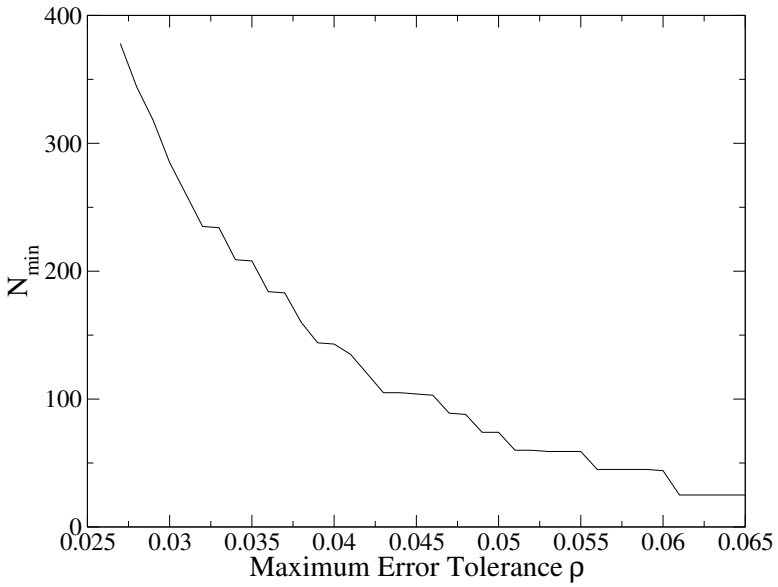


Figure 34: N_{min} as a function of ρ for $p = 0.9$ and $\epsilon = 0.05$.

To derive the previous probability, we need to make the additional assumption that the binary outcomes $I_i(R)$ of the N probing streams are independent. If that is the case, then $I(R, N)$ follows the binomial distribution with a success probability of $F_\tau(R)$. So, the probability that i out of N streams are of type-G is given by

$$P[I(R, N) = i] = \binom{N}{i} F_\tau(R)^i [1 - F_\tau(R)]^{N-i} \tag{28}$$

From Equations (27) and (28), we can calculate the minimum value N_{min} of streams required for a given error tolerance ρ and a given mis-classification probability ϵ. The probability $F_\tau(R)$ is determined based on the percentile that we aim to estimate. For instance, if we are interested in the 90% percentile then $F_\tau(R)$=0.9. Figure 34 shows N_{min} as a function of ρ for ϵ=0.05 and for estimating the 90% percentile. As expected, N increases quickly as we decrease the error threshold ρ. Specifically, as ρ becomes less than 4%, we need more than 100 samples (or probing streams).

In practice, generating a large number of probing streams increases the measurement overhead and it slows down the estimation process. Our objective in this work is to design a measurement tool that can track the avail-bw variation range in real time, even if the latter changes with time. For this reason, we prefer to use a relatively small number of probing

streams, even if the resulting error tolerance ρ is significant. Specifically, in the rest of the chapter we typically use $N = 40 - 50$ streams, limiting the maximum percentile error ρ to about 0.05-0.06.

4.3 Non-parametric estimation

In this section, we present a simple iterative algorithm for the estimation of the variation range $[A_\tau^L, A_\tau^H]$ in a given time scale τ. We refer to the following algorithm as *non-parametric*, in the sense that it does not assume a specific marginal distribution for the underlying avail-bw, or, equivalently, for the traffic at the tight link.

4.3.1 Algorithm

Suppose that we want to first estimate the higher bound A_τ^H of the variation range. If A_τ^H is the p-th percentile, then $p{=}F_\tau(A_\tau^H)$. The basic idea in the following algorithm is to iteratively adjust the probing rate R so that, based on Proposition 1, the fraction of probing streams that are of type-G is approximately p.

Specifically, in the n-th iteration of the algorithm the sender transmits N streams of rate R_n to the receiver. The receiver classifies each stream as type-G or type-L, and calculates the fraction $f_n = I(R_n, N)/N$ of streams that are of type-G. Based on Proposition 1, the expected value of f_n is equal to $F_\tau(R_n)$. So, if the rate R_n is close to the target percentile A_τ^H, we expect that f_n would be approximately equal to p. Similarly, if R_n is larger than A_τ^H then f_n is expected to be higher than p, while if R_n is less than A_τ^H then f_n is expected to be lower than p. The information about f_n is delivered back to the sender, which then sets the probing rate R_{n+1} accordingly.

In more detail, if f_n is within $p \pm \rho$, where ρ is a maximum percentile error, the rate R_n is reported as an estimate of the p-th percentile and the probing rate remains the same, i.e., $R_{n+1} = R_n$. If $f_n > p + \rho$, the sender needs to reduce the probing rate. Similarly, if $f_n < p - \rho$, the sender needs to increase the probing rate. The probing rate ratio R_{n+1}/R_n in the next iteration is based on the difference $f_n - p$. This is just a heuristic, but it is reasonable given that we do not have additional information about the shape of the underlying avail-bw distribution.

To avoid strong oscillations, we impose an upper bound on the rate variation between two successive iterations through a parameter b. A larger value of b allows faster convergence, especially under non-stationary conditions, but it also increases the estimation error. As will be shown later, a value of b around 0.10-0.20 is a good trade-off between accuracy and responsiveness, at least based on our validation experiments.

Algorithm 4.3.1 shows the pseudo-code of the non-parametric algorithm. The input parameters are the number of streams N, the probability p that corresponds to the desired percentile, and the error tolerance ρ. To measure the variation range $[A_\tau^L, A_\tau^H]$, the algorithm is executed twice in each iteration: N streams with probing rate R^H to estimate A_τ^H ($p = p^H$) and another set of N streams with rate R^L to estimate A_τ^L ($p = p^L$). The two sets of streams can be interleaved so that the reported estimates of the variation range cover the same time interval.

The non-parametric algorithm is iterative, and so it will be unable to track the avail-bw variation range if the latter does not remain roughly constant during at least a few iterations. The total probing duration for each iteration of the previous algorithm is $2N(\tau + T_{idle})$, where T_{idle} is the idle time which may be introduced between successive streams to reduce intrusiveness. For N=50, τ=20msec, and T_{idle}=80msec, two successive iterations of the previous algorithm will sample the same avail-bw distribution as a long as the underlying avail-bw process remains stationary for at least 10 seconds.

Algorithm 4.3.1: NON-PARAMETRIC(N, p, ρ)

repeat

Send N streams of duration τ at rate R_n

$I(R_n, N) \leftarrow 0$

for $i \leftarrow 1$ **to** N

do **if** $stream[i] =$ *type-G*
then $I(R_n, N) \leftarrow I(R_n, N) + 1$

$f_n \leftarrow I(R_n, N)/N$

if $f_n > p + \rho$

then $diff \leftarrow \text{MIN}(b, f_n - p)$
$R_{n+1} \leftarrow R_n * (1 - diff)$

else if $f_n < p - \rho$

then $diff \leftarrow \text{MIN}(b, p - f_n)$
$R_{n+1} \leftarrow R_n * (1 + diff)$

else $R_{n+1} \leftarrow R_n$
output R_n

4.3.2 Estimation with non-stationary load in single-hop path

In this section, we show examples of how the previous algorithm performs in a single-hop path with non-stationary traffic load that includes level shifts and short spikes. To make sure that the traffic load is realistic, we use packet traces captured by NLANR-MOAT at various OC-3 links (BWY-1063326722-1, COS-1049166362 and BWY-1063304167-1) [63]. Since we know the actual traffic load, we can calculate the exact 10%-90% percentiles of

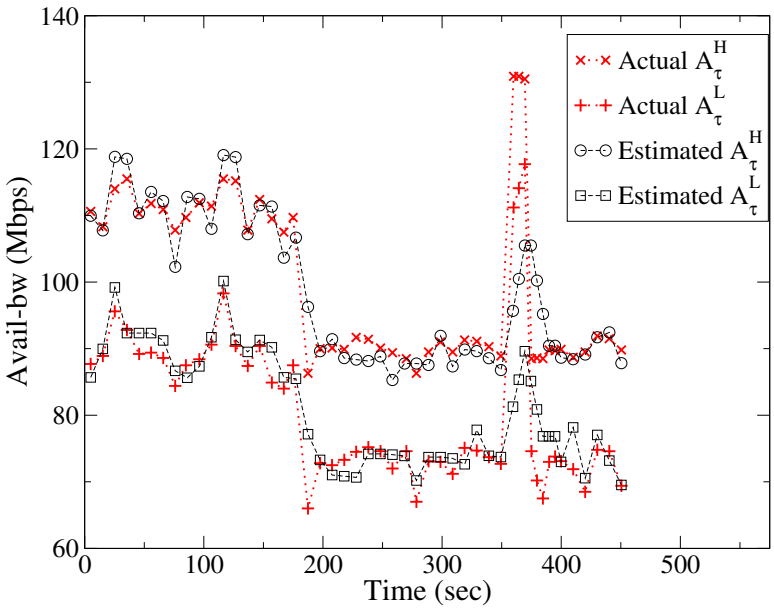


Figure 35: Actual and estimated variation range when b=.05

the avail-bw distribution, and so we can validate the previous estimation algorithm. In the following, the measurement time scale τ is 100msec. In the experiments of this section we make sure that the the classification of streams in type-G or type-L is always correct, by comparing the actual avail-bw in each probing interval with the probing rate.

To create non-stationary traffic loads, we merge different NLANR traces. Each trace is 90sec long, while the avail-bw process in each trace is stationary. The non-stationary traffic time series shown in Figure 35 and 36 was composed as follows: trace-1 was "played-back" twice, followed by trace-2 twice, followed from 20sec of trace-3 (to create the spike that occurs at t=360sec), and finally 50sec of trace-2 again.

The time series of the actual 10%-90% variation range was measured by segmenting the traffic trace in successive intervals of length $2N\tau$. At each segment, we calculate the empirical CDF of the avail-bw measured in time intervals of length τ. So, each successive interval of length $2N\tau$ results in a single measurement of the actual variation range in the time scale τ. The corresponding estimated variation range is inferred from the previous non-parametric algorithm using the same measurement time scale (τ) and measurement period ($2N\tau$).

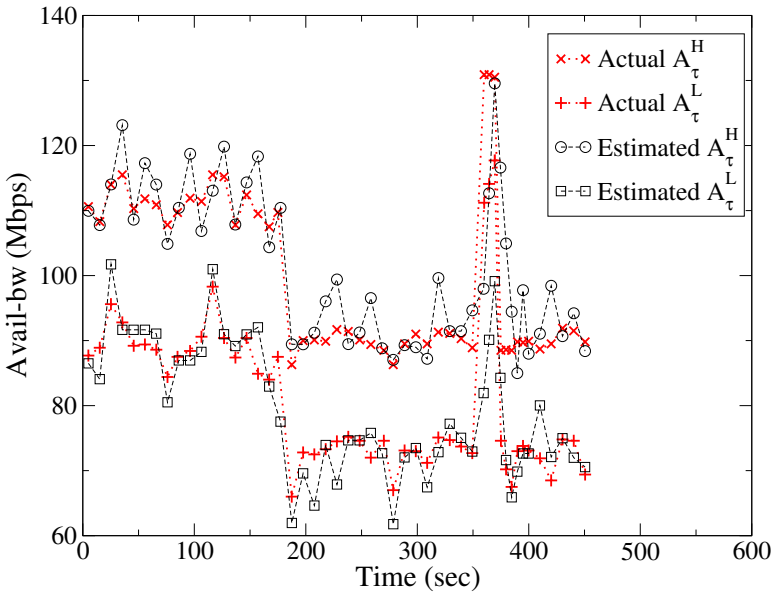


Figure 36: Actual and estimated variation range when b=.15

Figures 35 and 36 show the actual and the estimated 10%-90% variation range for two values of b. Notice that, overall, the estimation algorithm is able to successfully track the avail-bw variation range. During stationary time periods, the estimation error is less than 5%. The estimation errors are larger, however, during level shifts and short spikes.

The accuracy and responsiveness of the algorithm depend on b. The parameter b determines the maximum allowed rate variation in two successive iterations. When the avail-bw process is stationary, and the estimated variation range is already close to the actual variation range, a lower b performs better because it causes lower oscillations around the actual percentiles. For instance, the Root Mean Square Error (RMSE) of the estimated A_τ^H during the first 180 sec of the trace in Figure 35 is 2.4 for b=0.05 and 4.4 for b=0.15. The RMSE values for A_τ^L are 2.1 and 3.6, respectively.

On the other hand, a higher value of b is better during initialization, or when the traffic load exhibits frequent level shifts or spikes. For instance, notice the spike that occurs in the time interval [360,390] in Figures 35 and 36. Such an intense traffic spike can be due to a route flap or some form of network anomaly. With b=0.05 the estimation algorithm does not track successfully the magnitude of the traffic spike, while with b=0.15 the algorithm is

much more responsive. We note that the selection of b should be made based on the nature of the network path that is to be monitored. As it happens with most estimation tools, their accuracy depends on the calibration of certain parameters in the specific environment where these tools are to be used.

4.4 Parametric estimation

In this section, we present a parametric estimation algorithm that is based on the assumption that the avail-bw marginal distribution is Gaussian. This is a reasonable assumption for links with a large degree of flow multiplexing (high "vertical aggregation") and for sufficiently long measurement time scales τ (high "horizontal aggregation"). The Gaussian assumption in the context of network traffic and the required degrees of vertical and horizontal aggregation have been examined in [46] and the references therein. Specifically, the measurements presented in [46] show that the vertical aggregation of at least 25 users, with an aggregate average traffic rate of 25Mbps, is a good fit with the Gaussian model in time scales that are longer than 128msec. Also, the Gaussian model is a good approximation when the measurement time scale is longer than 1sec and the aggregate average rate is as small as a few Mbps. When it is not likely that the previous conditions hold, the non-parametric algorithm of the previous section should be used instead.

4.4.1 Algorithm

A Gaussian distribution is completely described by its mean and variance. Furthermore, the knowledge of any two percentiles of the Gaussian distribution is sufficient to compute the mean and variance. The basic idea in the following algorithm is to estimate two arbitrary percentiles of the avail-bw distribution based on Proposition 1. Then, we use these two percentiles to estimate the mean and variance of F_τ, and finally we estimate the user-specified variation range $[A_\tau^L, A_\tau^H]$.

In more detail, suppose that the avail-bw distribution has mean μ and variance σ_τ^2 in the time scale τ. Exactly as in the non-parametric algorithm, the sender generates N probing streams of rate R_1 and then it calculates the fraction f_1 of streams that are of type-G. Based on Proposition 1, the expected value of this fraction is equal to the cumulative

probability $F_\tau(R_1)$ that corresponds to rate R_1. So, if N is sufficiently large, we expect that $f_1 \approx F_\tau(R_1)$. The previous process is repeated for a different probing rate R_2, resulting in an additional constraint $f_2 \approx F_\tau(R_2)$. With the previous two constraints, we can then calculate the standard deviation and the mean of F_τ as follows:

$$\sigma_\tau = \frac{R_1 - R_2}{\phi^{-1}(f_1) - \phi^{-1}(f_2)} \tag{29}$$

$$\mu = R_1 - \sigma_\tau \phi^{-1}(f_1) \tag{30}$$

where ϕ^{-1} is the inverse of the standard normal distribution CDF. Finally, the percentiles that correspond to the variation range are:

$$A_\tau^H = \mu + \sigma_\tau \phi^{-1}(p^H) \tag{31}$$

$$A_\tau^L = \mu + \sigma_\tau \phi^{-1}(p^L) \tag{32}$$

It is important to note that the probing rates R_1 and R_2 need not be equal to A_τ^H or A_τ^L, respectively. Instead, it is sufficient to choose R_1 and R_2 so that the corresponding percentiles p_1 and p_2 are significantly different, i.e., $|p_1 - p_2| > \rho$. Furthermore, we can choose R_1 and R_2 so that they are at the left half of the Gaussian distribution. Doing so reduces the intrusiveness of the measurements, because the probing streams are of lower rate than the average avail-bw.

In practice, the probing rates R_1 and R_2 can be chosen to track two low percentiles, say the 20% and the 40%. This can be achieved by adjusting the two rates at the end of each repetition of the algorithm, based on the estimated Gaussian distribution. Notice that even with this optimization, the parametric algorithm remains non-iterative because the estimate of the variation range in each repetition of the algorithm does not depend on the estimate in the last repetition.

The pseudo-code for the parametric algorithm is given in Algorithm 4.4.1. As in the non-parametric algorithm, the transmission of the N streams of rate R_1 can be interleaved with the streams of rate R_2.

Algorithm 4.4.1: PARAMETRIC(N, p^H, p^L, p_1, p_2)

repeat

Send N streams of duration τ at rate R_1

Send N streams of duration τ at rate R_2

$I(R_1, N) \leftarrow 0$

for $i \leftarrow 1$ **to** N

do { **if** $stream[i, R_1] =$ *type-G*

then $I(R_1, N) \leftarrow I(R_1, N) + 1$

$f_1 \leftarrow I(R_1, N)/N$

$I(R_2, N) \leftarrow 0$

for $i \leftarrow 1$ **to** N

do { **if** $stream[i, R_2] =$ *type-G*

then $I(R_2, N) \leftarrow I(R_2, N) + 1$

$f_2 \leftarrow I(R_2, N)/N$

$\sigma_\tau \leftarrow \frac{R_1 - R_2}{\phi^{-1}(f_1) - \phi^{-1}(f_2)}$

$\mu \leftarrow R_1 - \sigma_\tau \phi^{-1}(f_1)$

$A_\tau^H \leftarrow \mu + \sigma_\tau \phi^{-1}(p^H)$

$A_\tau^L \leftarrow \mu + \sigma_\tau \phi^{-1}(p^L)$

$R_1 \leftarrow \mu + \sigma_\tau \phi^{-1}(p_1)$

$R_2 \leftarrow \mu + \sigma_\tau \phi^{-1}(p_2)$

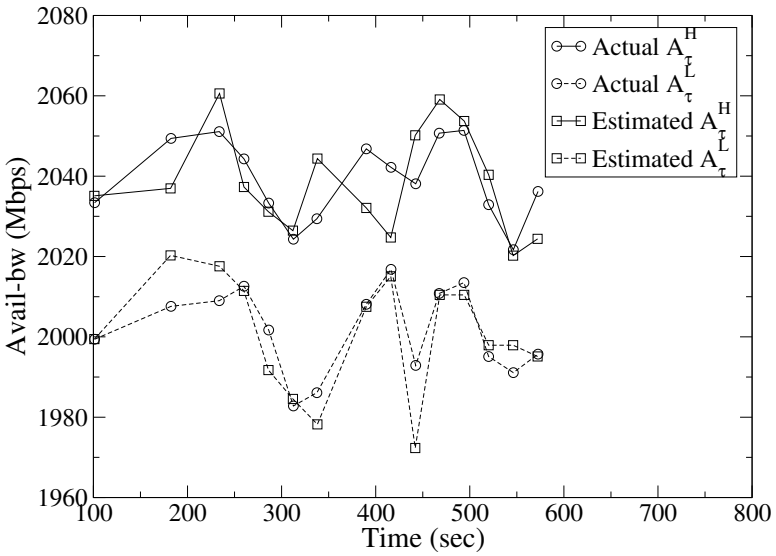


Figure 37: Estimated and actual 10-90% variation range with Gaussian traffic.

4.4.2 Validation examples

In this section, we illustrate the accuracy of the parametric algorithm for both Gaussian and non-Gaussian avail-bw distributions. We are again interested in the 10%-90% variation range. The probing rates are chosen based on the 10% and 40% percentiles, i.e., p_1=0.10 and p_2=0.40. The actual variation range is measured by calculating the empirical CDF in successive time windows of length $2N\tau$, as described in § 4.3.2.

Figure 37 shows the actual and the estimated variation range for an OC-48 NLANR packet trace (IPLS-CLEV-20020814-091000-1). The measurement time scale is τ=250msec. To examine whether the avail-bw distribution is Gaussian in that time scale, we calculate the kurtosis and skewness of the corresponding distribution. A Gaussian random variable has a skewness of zero and a kurtosis of 3. In this trace, the skewness and kurtosis are 0.21 and 3.15, respectively, meaning that the avail-bw distribution is reasonably close to Normal even though it is not a perfect match. The main observation in Figure 37 is that the parametric algorithm can closely track the variation range within 5% or better. The RMSE for this trace is 1.1.

On the other hand, Figure 38 shows the actual and estimated variation range for an OC-3 packet trace (ANL-1070464136) that deviates significantly from the Gaussian model.

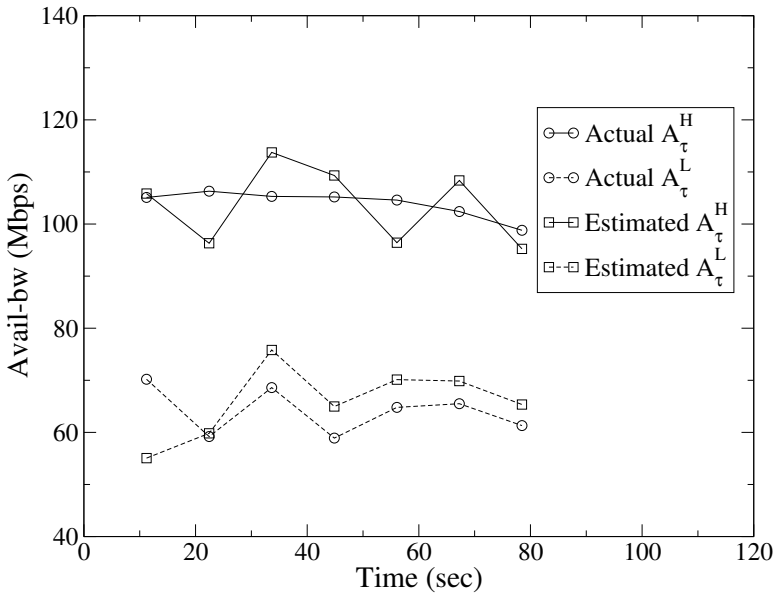


Figure 38: Estimated and actual 10-90% variation range with non-Gaussian traffic.

The measurement time scale in this case is τ=50msec, while the skewness and kurtosis are 0.46 and 5.23, respectively. Although the parametric algorithm is still able to track the variation range, there is a non-negligible bias in the estimation of the lower percentile, and the estimation error is significantly larger (RMSE=4.9) compared to the case of Gaussian traffic.

4.5 Pathvar

We have implemented both the non-parametric and parametric estimation algorithms in a tool called *Pathvar*. Pathvar consists of two components: the sender is responsible for transmitting the probing streams, while the receiver analyzes the One-Way Delays (OWDs) in each stream and determines whether a stream is of type-G or type-L. The two peers use a TCP connection to reliably transfer control messages and UDP datagrams for the probing streams. The number of streams N, stream duration τ, and the avail-bw variation range probabilities (p^L, p^H) are the key Pathvar inputs.

N determines the number of probing streams of a certain rate R that the sender will transmit to the receiver in each iteration. As described in § 4.2.2, N determines the accuracy with which we can infer the probability $F_\tau(R)$ that corresponds to the probing rate R.

Based on the results of § 4.2.2, Pathvar uses $N = 40$ targeting for an error tolerance $\rho = 0.05 - 0.06$ in the estimation of $F_\tau(R)$. The stream duration τ determines the measurement time scale for the average avail-bw, and it has to be chosen by the user based on the application requirements. We note however that τ should not be more than a few hundreds of milliseconds. The reason is that high-rate probing streams that last for too long can be network intrusive, causing congestion and packet losses.

Pathvar sends a total of $2N$ streams in each iteration: N streams at each of two different rates (as described in Sections 4.3 and 4.4). A stream is sent only when the previous stream has been acknowledged by the receiver, meaning that the duration of each iteration is $2N(\tau + RTT)$, where RTT is the Round-Trip Time between the two peers. Additionally, the probing streams of the two rates are interleaved, i.e. a stream of rate R_1 is followed by a stream of rate R_2, so that Pathvar probes the avail-bw distribution with the two rates almost simultaneously. After all $2N$ streams are received, the receiving peer examines whether each stream is of type-G or type-L, as described in [36], based on the detection of an increasing trend in the one-way delays of the probing packets.

Pathvar invokes either the non-parametric or the parametric algorithm depending on the specified time scale τ. If the latter is larger than 100msec, we prefer to use the parametric algorithm for three reasons. First, based on the measurement results of [46], we expect that in those time scales the avail-bw process will be sufficiently close to Normal. Second, with large values of τ, and consequently with long probing streams, the parametric algorithm gives us the advantage that we can select lower probing rates, reducing the intrusiveness of the measurements. Third, the non-parametric algorithm is not iterative and so it is less dependent on the stationarity assumption; that assumption can be questioned when τ is large.

In Pathvar, the sender timestamps each probing packet just before transmission. Upon arrival, the receiver records the arrival time and measures the OWD. The measured OWD differs from the actual OWD due to the clock offset between the two measurement peers. However, since we are only interested in the OWD differences, the clock offset does not affect the measurements as long as it is constant. The presence of clock skew does not affect

Pathvar because the stream duration is less than a second, while the typical magnitude of clock skew in modern quartz clocks is in the order of only a few microseconds per second. Context switching is another source of errors in the OWD measurements because buffering of packets in the kernel adds a variable delay component in the measured OWDs. Pathvar implements simple techniques to detect context switching and remove its effects, similar with the techniques developed for Pathload [32]. Finally, in the current version of Pathvar, the initial probing rates have to be provided by the user based on past experience with the measured path.

4.5.1 Testbed examples

We have evaluated the accuracy of Pathvar with both simulations and testbed experiments. The tight link at the testbed is a Fast Ethernet segment between two switches. The traffic at the tight link is generated by replaying the aggregate packet stream observed in NLANR traces. So, the packet sizes and interarrivals are based on realistic Internet traffic. To create non-stationary traffic conditions, and in particular level shifts, we concatenate traces with significantly different load.

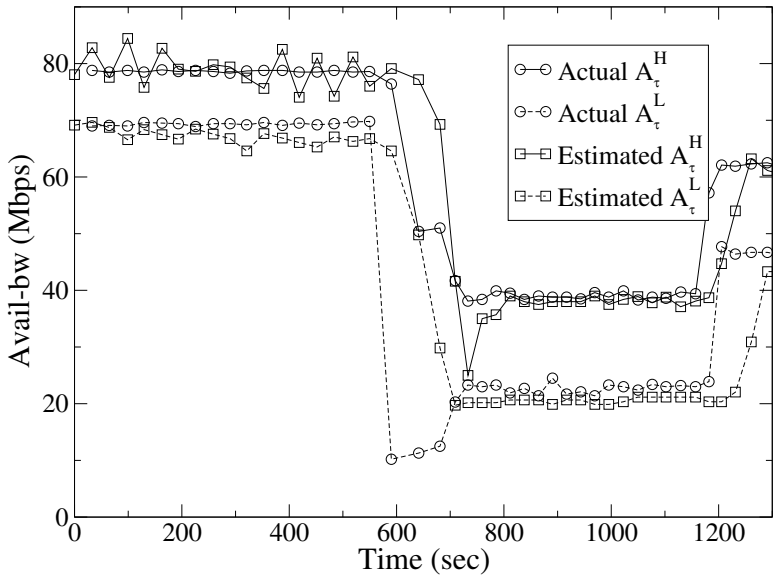


Figure 39: Pathvar experiment with non-parametric algorithm.

Figure 39 shows the actual and estimated 10%-90% percentile range, measured in a

time scale τ=40msec, for a non-stationary traffic load. The estimation in Figure 39 is performed with the non-parametric algorithm (b=0.2). A first observation is that, during the stationary epochs, Pathvar tracks the actual variation range within 10% or better. A second observation is that after level-shift events, the non-parametric estimation algorithm needs some considerable amount of time (100-200sec) to reconverge to the correct variation range. This delay can be reduced by using a larger value of b, but with an associated cost in the accuracy of the estimation during stationary periods. A future improvement that we consider is to dynamically increase b, upon the detection of frequent level-shifts or other forms of non-stationarity, and to gradually decrease b when the avail-bw remains at the same level.

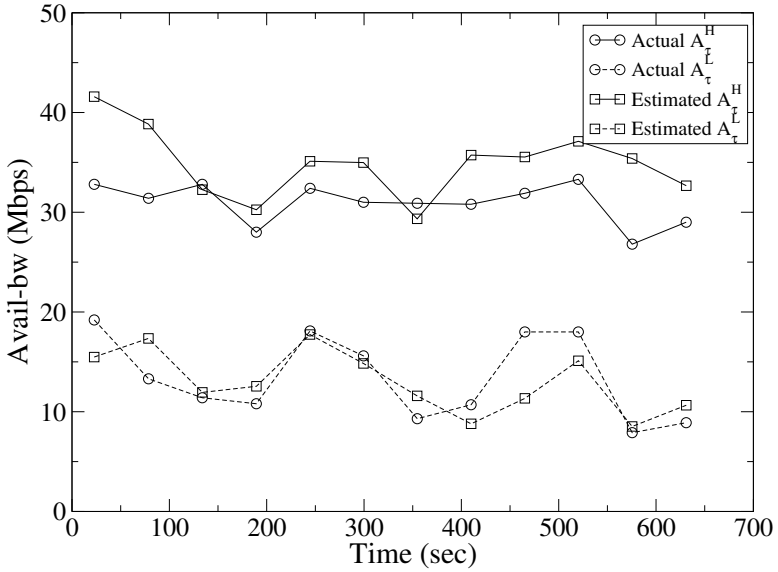


Figure 40: Pathvar experiment with parametric algorithm.

In Figure 40, we use the parametric algorithm instead. The actual and estimated 10%-90% percentile range are measured in a time scale τ=250msec. The traffic load is non-stationary (generated from replaying multiple times a 90-sec NLANR trace), but with a marginal distribution that is quite close to the Gaussian model.

A first observation is that the tool needs about $2N(\tau + RTT)$=55 seconds to generate each estimate of the variation range. Notice that this large latency is not an intrinsic characteristic of the parametric algorithm, but it is due to the large measurement time scale

τ and the associated long duration of each probing stream. Second, the estimation error is in the order of 10-20%. The reader should not conclude that the parametric algorithm is less accurate than the non-parametric algorithm. In general, the accuracy of the two algorithms is comparable when they are both applied on the same traffic and with the same value of τ, as long as the traffic process is stationary and Gaussian.

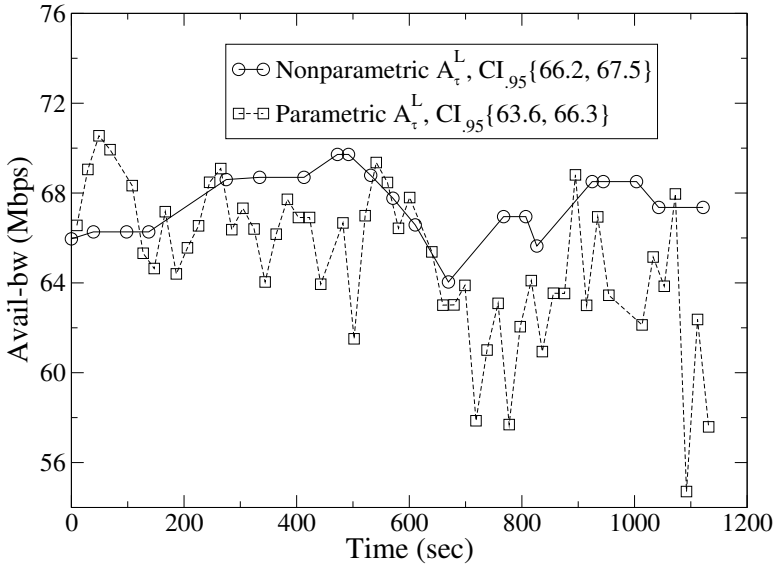


Figure 41: Non-parametric and parametric estimates of 20% percentile for τ=40msec.

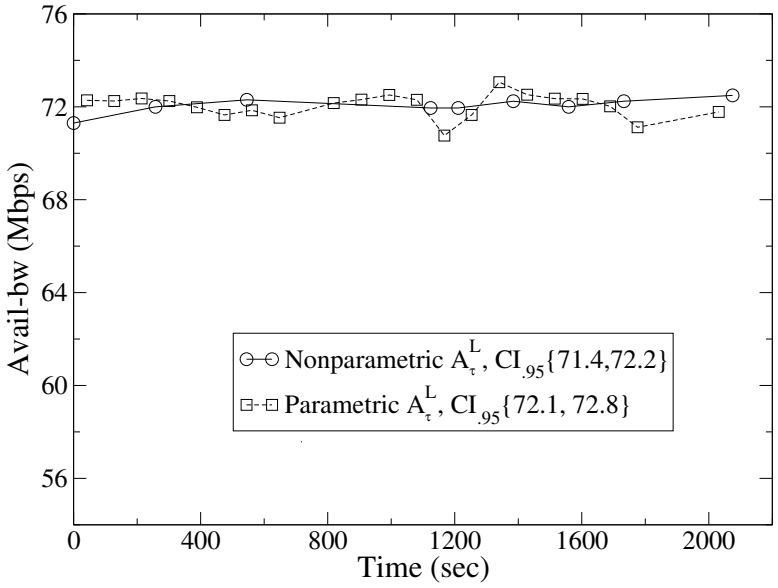


Figure 42: Non-parametric and parametric estimates of 20% percentile for τ=140msec.

Next, we compare the accuracy of the parametric and non-parametric algorithms under the same traffic load. In the following graphs, we show a single avail-bw percentile, rather than a variation range, to avoid cluttering. For each algorithm, we show the time series of the 20-th percentile avail-bw estimates, as well as the 95-th Confidence Intervals (CI) of those estimates.

Figures 41 and 42 show the effect of the measurement time scale τ on the accuracy of the two algorithms. The traffic is generated by replaying the NLANR trace MRA-10621841. The 20-th percentile of the avail-bw distribution during the entire trace is 67.0Mbps at τ=40msec, and 72.1Mbps at τ=140msec. The non-parametric algorithm estimates this percentile quite accurately in both measurement time scales. The parametric algorithm, on the other hand, is accurate when τ=140msec, but it underestimates the given percentile when τ=40msec. The reason is that in that shorter measurement time scale, the traffic deviates significantly from the Normal distribution.

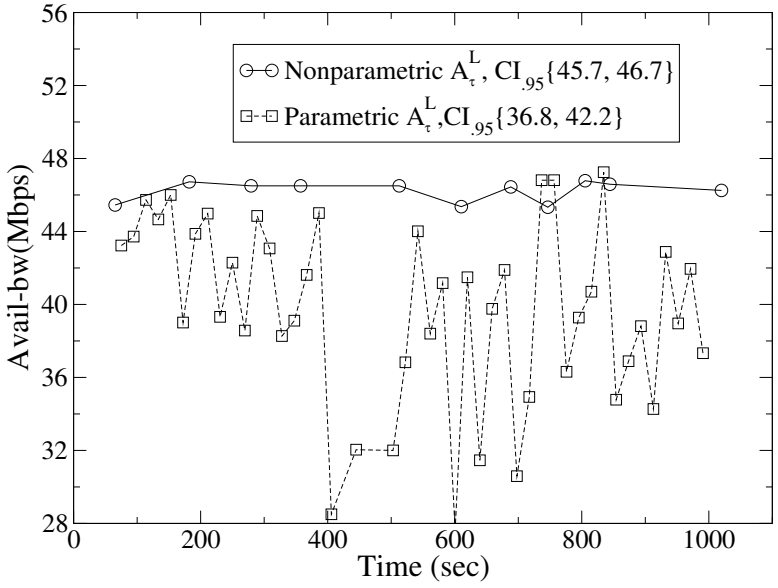


Figure 43: Non-parametric and parametric estimates of 20% percentile for low vertical aggregation.

Figures 43 and 44 show the effect of the degree of statistical multiplexing ("vertical aggregation") on the accuracy of the two algorithms. To generate traffic with a lower degree of multiplexing we replay 20 large flows extracted from an NLANR trace, and to

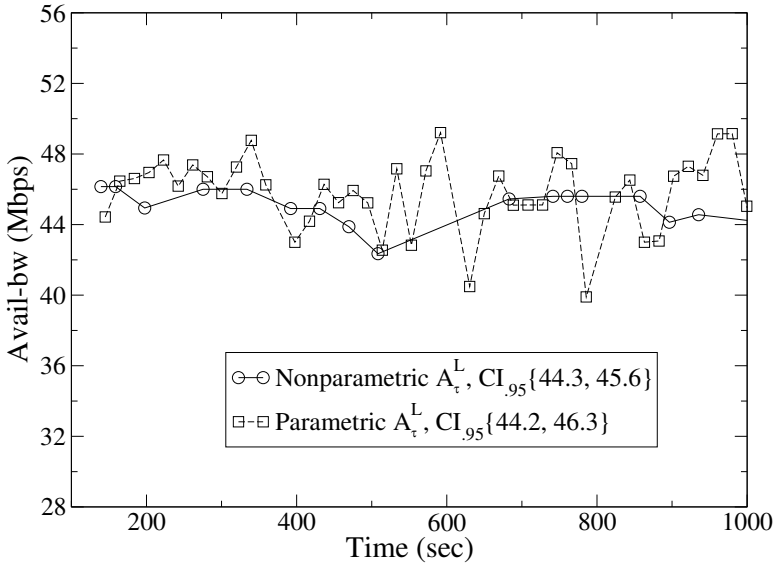


Figure 44: Non-parametric and parametric estimates of 20% percentile for high vertical aggregation.

generate traffic with higher multiplexing we replay approximately 3000 smaller flows from the same trace. The 20-th percentile of the avail-bw distribution during the entire trace is 45.9Mbps and 44.9Mbps, respectively. In both cases, the average traffic rate (and avail-bw) is about the same. The non-parametric algorithm produces accurate estimates of 20-th percentile with both degrees of multiplexing. The parametric algorithm, on the other hand, is accurate only when the traffic is highly aggregated. The reason is that in the latter the traffic deviates significantly from the Normal distribution.

To summarize the experiments of this section, the parametric algorithm performs better than the non-parametric algorithm under non-stationary conditions (especially level shifts and traffic spikes) because it is not iterative. On the other hand, if the traffic is not Gaussian because of low horizontal or vertical aggregation, then the non-parametric algorithm performs better. Obviously, the accuracy of Pathvar is worse when both previous assumptions do not hold, i.e., with non-stationary and non-Gaussian traffic. This may be the case in paths where the tight link is the host network interface or a LAN link. In such environments, the traffic load is sporadic, generated by only a few high-throughput flows, and so the resulting avail-bw process can be both non-stationary and non-Gaussian.

4.5.2 Internet Experiments

We have also used Pathvar to measure the avail-bw variation range in several Internet paths. The objective of these experiments is not to perform validation, given that we do not know the actual avail-bw distribution, but to observe how the avail-bw variation range changes with time in real Internet paths. In this section, we show some preliminary results from two Internet paths between Georgia Tech (in Atlanta GA) and two universities in Greece (in Ioannina and Heraclion-Crete). In both cases, we have evidence that the tight link is the campus access link of the Greek universities (based on the corresponding MRTG graphs).

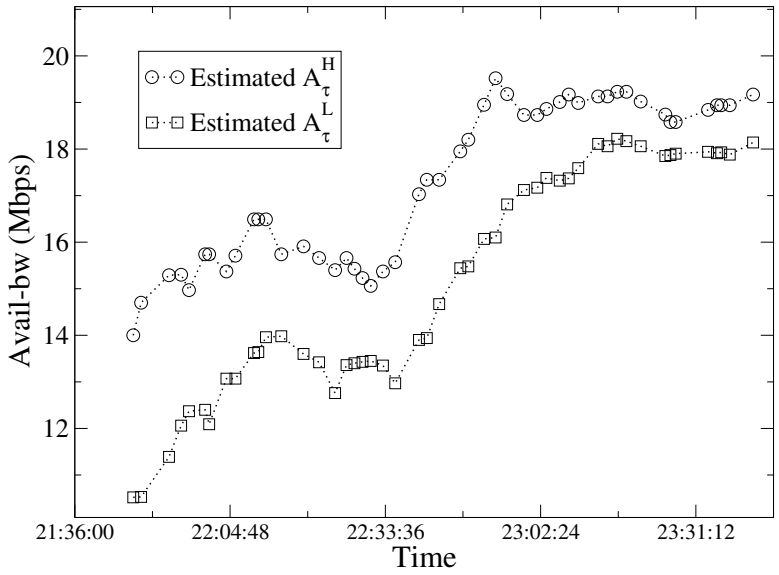


Figure 45: Variation range estimates at the Internet path from Georgia Tech to University of Ioannina.

Figure 45 shows the estimated 20%-80% variation range of the avail-bw for the path from Georgia Tech to University of Ioannina over a two-hour time period. The time reported in the x-axis refers to local time in Greece. The measurement timescale is 40msec and the estimates are obtained using the non-parametric algorithm. A first observation is that the avail-bw gradually increases, especially after 240pm. A second, more interesting observation is that the variation range decreases as the average avail-bw increases. We discuss the relation between avail-bw variability and tight link utilization in the next section.

Figure 46 shows the estimated 20%-80% variation range for the avail-bw for a path from University of Crete (UoC) in Heraclion to Georgia Tech over a three-hour window.

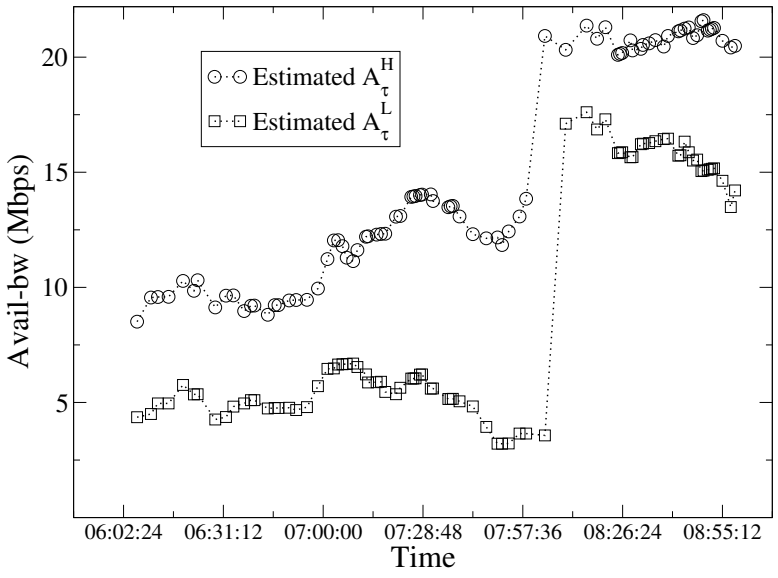


Figure 46: Variation range estimates at the Internet path from University of Crete to Georgia Tech.

The time reported in the x-axis refers to local time in Greece. The measurement timescale is 40msec and the estimates are obtained using the non-parametric algorithm. A first observation is that the avail-bw shows a sharp increase at 8:00am. We confirmed this unusual behavior with the MRTG graph of the UoC campus access link. One possible explanation is that certain applications (e.g., p2p file transfers) are blocked during working hours. Another interesting observation is that even though the average avail-bw is roughly constant between 6:00am and 8:00am, the variation range fluctuates significantly. This illustrates that estimating only the average avail-bw may be an insufficient indicator for the load of a network path.

4.6 Variability factors

The previous sections focused on the estimation of the avail-bw variation range through end-to-end measurements. Which are the factors, however, that affect the variability of the avail-bw distribution? Why does the traffic appear to be more "bursty" in some paths than in other paths? Two pieces of conventional wisdom are that "heavier load conditions also produce wider traffic variations" and that "a higher degree of multiplexing makes the traffic smoother". Under which conditions, however, are these statements true?

In this section, we focus on four different factors, and show how they affect the variability of the avail-bw distribution. These factors are the traffic load at the tight link, the number of competing flows, the rate of competing flows, and of course the measurement time scale. The first three factors are related to the traffic characteristics at the tight link, while the last factor is related to the way the avail-bw is measured. Even though these factors have been examined in different contexts before, our focus here is specifically on the way these factors affect the variation range of the avail-bw distribution.

The following results are based on a simulation study in which we measure the avail-bw variation range as we vary each of the previous four factors. Specifically, we have implemented an NS module for Pathvar that is identical to the actual prototype described in § 4.5. Unless noted otherwise, we use the following parameters in the simulation: τ=50msec and N=40 streams. The simulation topology includes a tight link with capacity C_t=50Mbps. The traffic at the tight link is generated by a large number of edge nodes, and it resembles short HTTP flows running over TCP NewReno. Each such flow transfers 10-15 packets from a server to a client through the tight link, sleeps for a random time interval (that is adjusted based on the desired average load), and then repeats the previous cycle. Note that each time when the flows wakes up from sleep, it goes through a slow start phase.

During each simulation Pathvar runs M=25 consecutive times, estimating a 10%-90% variation range $[A_\tau^L(i), A_\tau^H(i)]$, for $i = 1 \ldots M$, after each run. To summarize the M ranges into a single figure, we calculate the average width $\hat{V}$ of the estimated variation ranges as follows

$$\hat{V} = \frac{\sum_{i=1}^{M}(A_\tau^H(i) - A_\tau^L(i))}{M} \tag{33}$$

We also calculate the standard deviation $\hat{E}$ of these M samples, to quantify their dispersion around $\hat{V}$.

The following simulations also serve as a validation study of Pathvar. To do so, we collect a traffic trace at the tight link during the simulation and then measure the width $V = A_\tau^H - A_\tau^L$ of the actual variation range $[A_\tau^L, A_\tau^H]$. The comparison of $\hat{V}$ with V indicates whether Pathvar can successfully estimate the avail-bw variation range width.

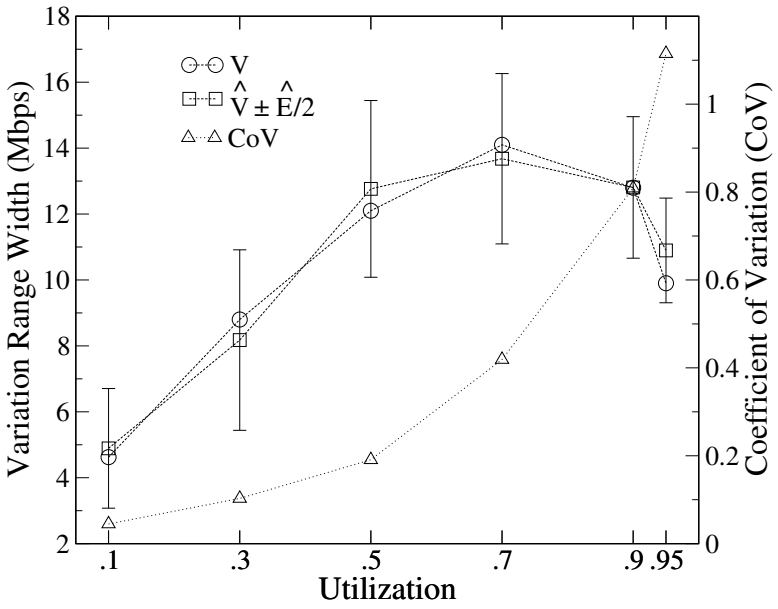


Figure 47: Effect of tight link utilization on the variation range width (left Y-axis) and CoV (right Y-axis).

4.6.1 Effect of tight link utilization

The first factor we consider is the average utilization u at the tight link. From queueing theory we know that the variance of the queueing delay or backlog in most queueing systems increases as the utilization increases [11]. How does the utilization affect the avail-bw variability however?

In Figure 47, we show the estimated and the actual variation range width for six values of u. The utilization is controlled by adjusting the number of TCP clients. The measurement time scale is τ=30msec. The first observation, in terms of validating Pathvar, is that the actual variation range width V is very close to the estimation $\hat{V}$, and always within a range of $\pm\hat{E}/2$.

Second, the variation range width increases with u up to a certain point. After that point, the variation range width decreases with u. The point of the maximum variability in this particular simulation occurs when the utilization is around 70%, but this depends on the measurement time scale and on the characteristics of the traffic mix. What is the reason for this non-monotonic variation of V with u? Intuitively, when u is relatively low,

i.e., in light load conditions, V increases with u because of the increasing variability in the offered load. With Poisson traffic, the variance of the offered load increases linearly with the average traffic rate. As u approaches 100%, however, the tight link often becomes saturated. During the time periods that the tight link is saturated, the departure rate at the output of the tight link remains constant, and so the avail-bw variability drops. In the extreme case that the tight link is always fully utilized, the avail-bw remains constantly zero, and so its variability is also zero. Note that it is important to distinguish between the traffic variability at the input of a link versus at the output. Even if the input rate has high traffic spikes, the traffic rate at the output is essentially "clamped" by the link capacity. It is this clamping effect that causes the variation range reduction at high loads. This effect has been also studied in [87], examining the relation between load and traffic variance.

It is interesting that the Coefficient of Variation (CoV) of the traffic at the output of the tight link follows a different trend than the variation range (see Figure 47). Specifically, the CoV, which is defined as the standard deviation of the avail-bw over the average avail-bw, increases monotonically with u. This trend should not be misinterpreted as an indication that heavier loads cause wider traffic variability. This is only true in relative terms, when the avail-bw variability is normalized by the average avail-bw. In absolute terms, instead, the avail-bw variability reaches its maximum when the link is significantly loaded but not congested.

4.6.2 Statistical multiplexing effects and scaling models

Another conventional wisdom is that a higher degree of statistical multiplexing, under the same load conditions, makes the traffic smoother. To examine the validity of this statement, we first need to clarify what it means to increase the degree of multiplexing at a link.

We distinguish between two scaling models. In the first, referred to as "capacity scaling", we increase the capacity of the tight link proportionally to the number of competing flows. The average rate of each flow, as well as the utilization of the tight link, remain constant. In the second, referred to as "flow scaling", we increase the number of competing flows at the tight link while proportionally decreasing their average rate; the capacity and utilization of

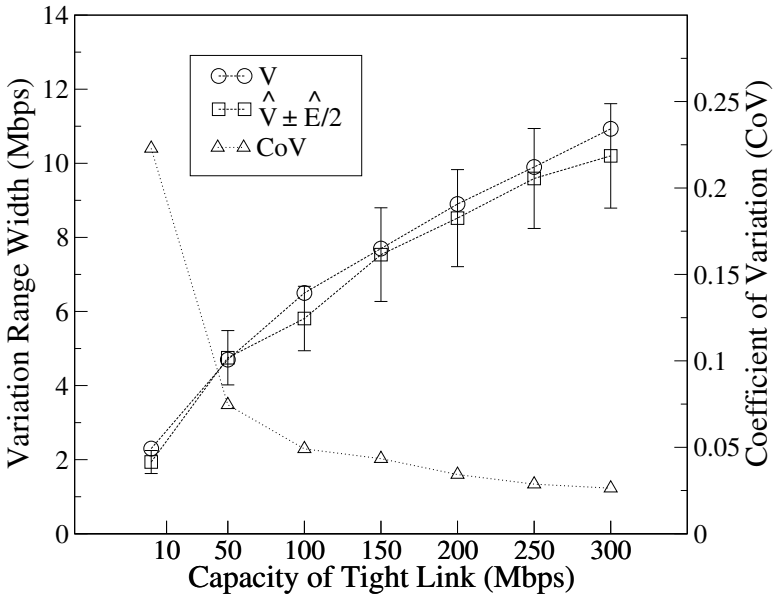


Figure 48: Effect of capacity scaling on the variation range width (left Y-axis) and CoV (right Y-axis).

the tight link remain constant. Note that in both scaling models the number of competing flows at the tight link increases, while the utilization of that link remains the same.

4.6.2.1 Capacity Scaling

To simulate capacity scaling, we increase the number of TCP clients U proportionally to the capacity C_t of the tight link. Specifically, U is increased from 3 to about 90, C_t is increased from 10Mbps to 300Mbps, while the tight link utilization is kept constant at 50%. Each TCP client transmits an average of 1000 packets, then sleeps for a random time interval between 2-5 secs, and then repeats the previous cycle. The capacity of the access link of each client is fixed to 2Mbps, and the average rate of each TCP flow also remains constant.

Figure 48 shows the effect of capacity scaling on the avail-bw variation range width and CoV. Interestingly, capacity scaling has a different effect on the avail-bw variability, depending on whether we look at the variation range width or at the CoV. The former increases with C_t, while the latter decreases. To understand why, suppose that X_i is the traffic process generated by flow i, while $Y = \sum_{i=1}^{U} X_i$ is the aggregate traffic process at the tight link generated by U flows. If the U flows are independent, which is a reasonable

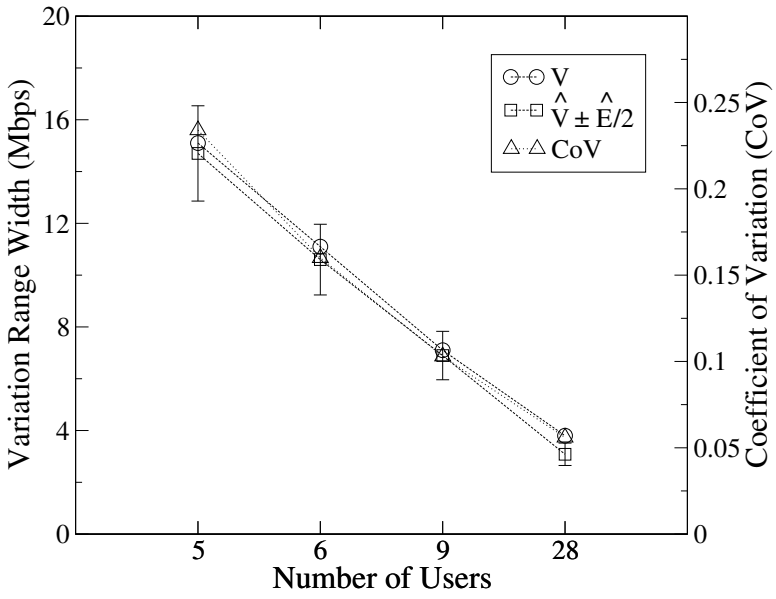


Figure 49: Effect of flow scaling on the variation range width (left Y-axis) and CoV (right Y-axis).

assumption when the tight link is not congested, and if we assume for simplicity that the flows are identically distributed with $\text{Var}[X_i] = \text{Var}[X]$, then we have that $\text{Var}[Y] = U\text{Var}[X]$. So, the variation range width of Y will increase with U. Actually, if Y is Gaussian, then the width V of a symmetric variation range is proportional to the standard deviation of Y.[2] In that case, V increases proportionally to $\sqrt{U}$. The CoV of the avail-bw, on the other hand, is equal to

$$\text{CoV} = \frac{\sqrt{C - U\text{Var}[X]}}{C - UE[X]} = \frac{\sqrt{U\text{Var}[X]}}{C - UE[X]} \tag{34}$$

In capacity scaling, U increases proportionally with C, and so the CoV decreases as $1/\sqrt{C}$. So, the relative variability of the avail-bw decreases with capacity scaling, even though the absolute width of the variation range increases.

4.6.2.2 Flow scaling

To simulate flow scaling, we increase the number of users U (TCP clients) decreasing proportionally the average traffic rate of each user. This rate reduction is achieved by decreasing

[2]For instance, if Y is Gaussian, then it is easy to calculate that the 10%-90% variation range width is equal to 2.56σ, where σ is the std-deviation of Y.

the capacity C_e of the edge link that connects each user to the tight link. The throughput of the TCP transfers is determined by C_e in these simulations. Figure 49 shows the effect of capacity scaling on the avail-bw variation range width and CoV. In the case of flow scaling, note that both the absolute variation range as well as the CoV decrease as the number of users increases.

An interesting question is, why does the variation range width decrease with flow scaling, but it increases with capacity scaling? Consider again the simple model of the previous paragraph. The variance of Y is $\text{Var}[Y] = U\text{Var}[X]$, assuming independence among the U users. The difference with capacity scaling, however, is that in flow scaling the variance $\text{Var}[X]$ of each flow decreases as U increases. This is at least the case for most traffic processes: their variance decreases as the average rate decreases. In the Poisson process, the variance is simply equal to the average rate. In the Poisson Pareto Burst Process [95], which creates self-similar traffic, the variance is proportional the square of the average rate. As long as $\text{Var}[X]$ decreases faster than the increase in U, the variance $\text{Var}[Y]$ will decrease as we increase the number of users. This is the case for the traffic mix that we simulate in Figure 49, or for the Poisson Pareto Burst Process. On other hand, this would not be the case for the Poisson process, in which $\text{Var}[Y]$ remains constant as we increase U.

The fact that the CoV decreases with flow scaling also depends on the relation between U and $\text{Var}[X]$. As in the previous paragraph, the avail-bw CoV is given by (34). Since the denominator (average avail-bw) remains constant, the CoV decreases if the variance of individual flows decreases faster than the flow average rate.

4.6.3 Effect of measurement timescale

As mentioned in the Introduction, the variability of the avail-bw decreases with τ. The rate of decrease, however, can be very different depending on the correlation structure of the underlying traffic process. If $A_\tau(t)$ is an IID random process, then the variance decreases inversely proportional with the length of the averaging time scale

$$\text{Var}[A_{k\tau}] = \frac{\text{Var}[A_\tau]}{k} \tag{35}$$

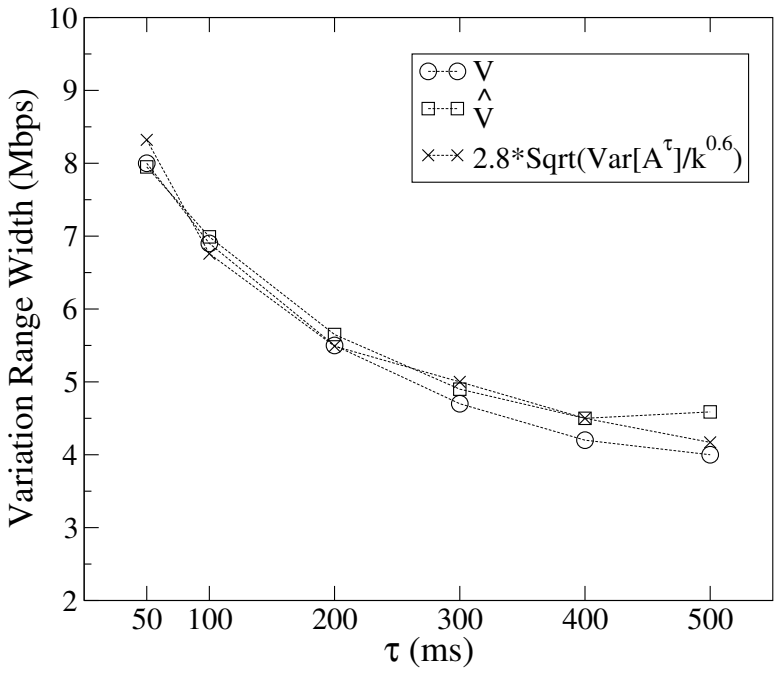


Figure 50: Effect of time scale τ on the variation range width.

On the other hand, if $A_\tau(t)$ is an exactly self-similar process with Hurst parameter $0.5 < H < 1$, the variance decreases slower

$$\text{Var}[A_{k\tau}] = \frac{\text{Var}[A_\tau]}{k^{2(1-H)}} \tag{36}$$

A tool such as Pathvar can estimate the variation range in different time scales. Consequently, it is possible to infer through end-to-end measurements whether the avail-bw process is an IID or a self-similar process, and in the latter, to measure the local Hurst parameter in a certain range of time scales.

Figure 50 shows the actual and the estimated variation range width of the avail-bw in six measurement time scales: τ=50, 100, 200, 300, 400, and 500msec. As we expected, V decreases with τ. More interestingly, however, the decrease rate is consistent with a self-similar process with Hurst parameter H=0.7. Of course this scaling behavior is valid locally in the previous range of τ; the Hurst parameter may be different in larger time scales.

4.7 Pathvar Overhead and Latency

In this section, we focus on the overhead and the measurement latency of both the algorithms implemented in Pathvar. We use the simulation setup as described in §4.6.

Figure 51 shows the measurement latency of Pathvar when the non-parametric algorithm is used to estimate the variation range. The measurement latency of the non-parametric algorithm depends on the number of iterations needed for converging to a avail-bw variation range estimate. The number of iterations increase with the cross traffic rate in the path, since it requires more iterations to converge to the thresholds used for the classification of a stream as type-G or type-L [36]. This explains the decreasing trend in the measurement latency as the avail-bw increases. As discussed in §4.3, the probing duration for each iteration is $2N(\tau + T_{idle})$. For the values of various parameters used in this simulation ($N = 50$, $\tau = 20msec$, and $T_{idle} = 80msec$), each iteration takes about 10 seconds. Based on 10sec per iteration, the number of iterations vary approximately from 15 to 11 in Figure 51.

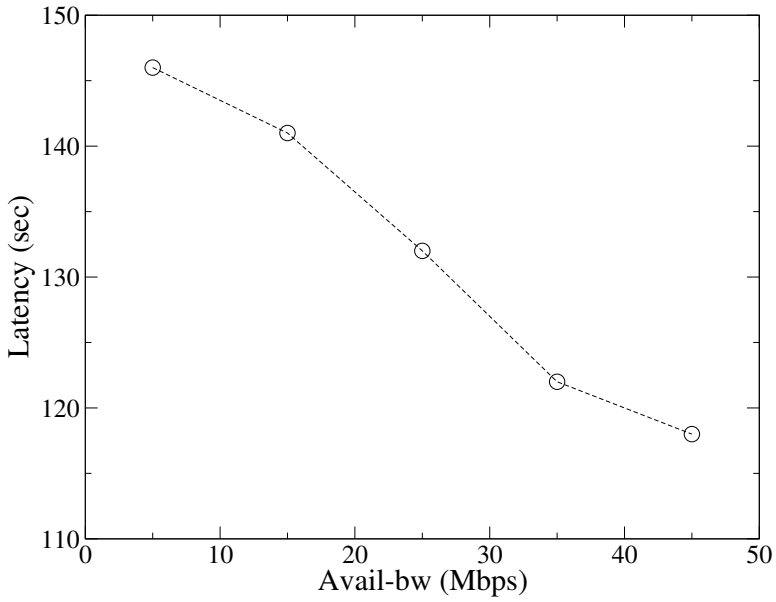


Figure 51: Measurement latency for the non-parametric algorithm.

The measurement latency for the parametric algorithm does not vary with avail-bw and is equal to $2N(\tau+T_{idle})$. This is because the parametric algorithm is non-iterative in nature.

Figures 52 and 53 show the overhead of Pathvar, expressed in terms of average sending rate, for the non-parametric and the parametric algorithm, respectively. In both cases, the average sending rate is less than 30% of the average avail-bw.

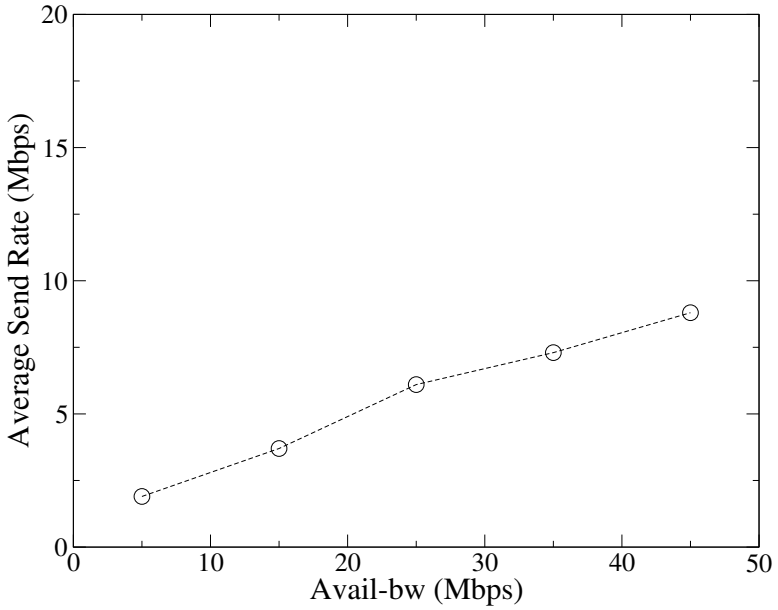


Figure 52: Average sending rate for the non-parametric algorithm.

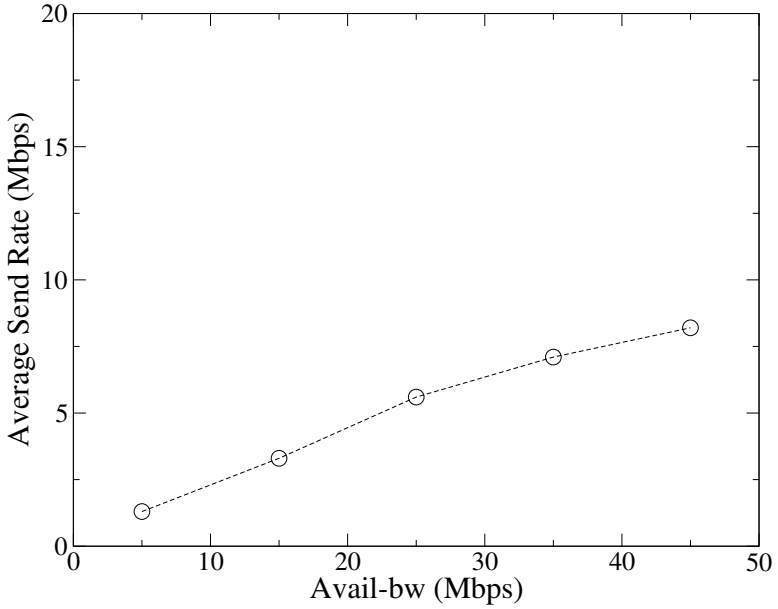


Figure 53: Average sending rate for the parametric algorithm.

4.8 Summary

This chapter focused on the estimation of the avail-bw variation range using end-to-end network measurements. To the extent of our knowledge, this is the first work that aimed to measure the variability of the avail-bw, rather than its mean. We developed and evaluated two complementary estimation algorithms. The selection among the two algorithms depends on the measurement time scale, the degree of multiplexing at the tight link, and the stationarity of the traffic at the measured path. The accuracy of the proposed algorithms will probably be satisfactory for most applications, with relative errors up to 10-20%.

CHAPTER V

EFFECTS OF INTERRUPT COALESCENCE ON NETWORK MEASUREMENTS

The arrival and departure of packets at a Network Interface Card (NIC) are two events that the CPU learns about through interrupts. An interrupt-driven kernel, however, can get in a *receive livelock* state, where the CPU is fully consumed with processing network interrupts instead of processing the data contained in received packets [61]. In Gigabit Ethernet (GigE) paths, 1500-byte packets can arrive to a host in every $12\mu s$. If the interrupt processing overhead is longer than $12\mu s$, receive livelock will occur when an interrupt is generated for every arriving packet.

The system overhead for each arriving packet consists of a context switch, followed by the execution of the network interrupt service routine, followed by another context switch. In order to reduce the per-packet CPU overhead, and to avoid receive livelock, most high-bandwidth network interfaces today (in particular GigE cards) use *Interrupt Coalescence* (IC). IC is a technique in which NICs attempt to group multiple packets, received in a short time interval, in a single interrupt. Similarly, at the sending host, the CPU learns about the departure of several packets through a single interrupt. IC is not a new technique. It becomes common whenever a new LAN technology brings the network speed closer to the CPU speed; for instance, it was also used in the early days of Fast Ethernet NICs [61].

In this work, we identify and describe the negative effects of IC on both active and passive network measurements. Specifically, IC can affect active bandwidth estimation techniques, when the latter ignore the possibility of IC at the receiver's NIC [40]. Bandwidth measurements are classified as either capacity estimation or available bandwidth estimation. For a recent survey, we refer the reader to [73]. Capacity estimation is based on the analysis of dispersion measurements from back-to-back packet pairs and packet trains. Available bandwidth estimation, on the other hand, is based on the relation between the departure

and arrival rates of periodic packet streams.

We have modified our previous capacity and available bandwidth measurement techniques, presented in [17] and [33] respectively, so that they can provide accurate estimates *in the presence of IC*. For capacity estimation, a sufficiently long packet train can provide a clear IC signature, in which packets are received in successive bursts at a certain time period. Based on this signature, we can detect IC, filter our the erroneous measurements, and estimate the correct path capacity. For available bandwidth estimation, we can similarly detect the IC signature in the one-way delay measurements of periodic packet streams. It is then simple to ignore the one-way delay measurements that have been affected by IC, and to determine whether the remaining measurements show an increasing trend.

IC can also affect passive measurements that use commodity NICs for packet trace capture. The reason is that IC can alter the packet interarrivals in the monitored traffic stream. We finally show that IC can be detrimental to TCP self-clocking, causing bursty delivery of ACKs to the sender and bursty transmission of data segments.

The rest of the chapter is structured as follows. §5.1 explains how IC works in two popular GigE controllers. §5.2 illustrates the negative effects of IC in both active and passive network measurements, and it describes how to filter dispersion and one-way delay measurements in the presence of IC.

5.1 Description of Interrupt Coalescence

The basic objective behind IC is to reduce the number of network interrupts generated in short time intervals (in the order of a few hundereds of microseconds). This can be achieved by delaying the generation of an interrupt for a few hundreds of microseconds, expecting that more packets will be received in the meanwhile. Most GigE NICs today support IC by buffering packets for a variable time period (*dynamic* mode), or for a fixed time period (*static* mode) [27, 82]. In *dynamic* mode, the NIC controller determines an appropriate interrupt generation rate based on the network/system load. In *static* mode, the interrupt generation rate or latency is set to a fixed value, specified in the driver.

Users (with root privileges) can modify the default controller behavior by changing

the parameters provided by the NIC drivers. For instance, the driver for the Intel GigE E1000 controller [28] provides the following parameters for adjusting the IC receive (Rx) and transmit (Tx) operations:

- *InterruptThrottleRate*: the maximum number of interrupts per second. It is implemented by limiting the minimum time interval between consecutive interrupts.
- *(Rx/Tx)AbsIntDelay*: the delay between the arrival of the first packet after the last interrupt and the generation of a new interrupt. It controls the maximum queueing delay of a packet at the NIC buffers.
- *(Rx/Tx)IntDelay*: the delay between the last arrival of a packet and the generation of a new interrupt. It controls the minimum queueing delay of a packet at the NIC buffers.

Note that the previous parameters can be combined to meet constraints on the maximum interrupt rate and on the maximum/minimum IC-induced queueing delays. In case of conflict, *InterruptThrottleRate* has a higher precedence than *(Rx,Tx)AbsIntDelay* and *(Rx,Tx)IntDelay* parameters.

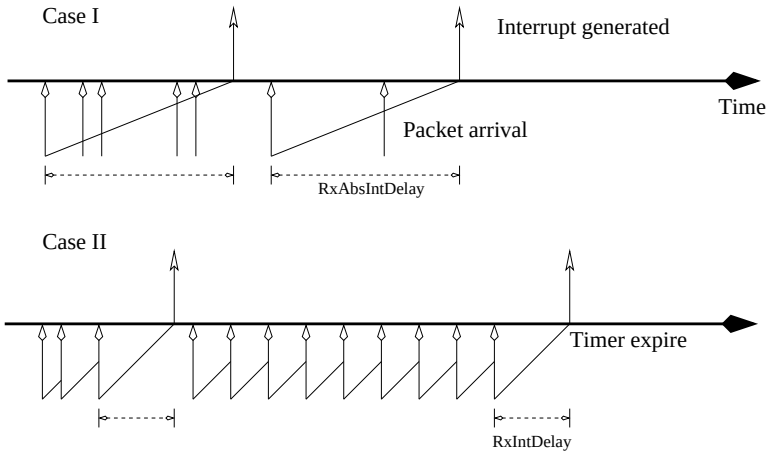


Figure 54: Illustration of IC with *RxAbsIntDelay* (Case I) and with *RxIntDelay* (Case II).

Figure 54 illustrates the interrupt generation process at the receiving host with only the *RxIntDelay* or *RxAbsIntDelay* constraint. In case I, the *RxAbsIntDelay* timer is set to expire after a certain delay since the arrival of the first packet after the last interrupt. All

the subsequent packets are buffered at the NIC buffers, and they are delivered with the next interrupt. If the minimum packet spacing at the given NIC is T, the maximum number of packets that may need to be buffered is $RxAbsIntDelay/T$. In case II, every packet arrival resets the *RxIntDelay* interrupt timer. Consequently, if packets arrive periodically with a lower dispersion than *RxIntDelay*, the interrupt generation can be delayed indefinitely, causing NIC buffer overflows.

The SysKonnect GigE NIC [83] provides two parameters: *moderate* and *intpersec*. *Moderate* determines whether IC is enabled, and whether it is in *static* or *dynamic* mode. *Intpersec* determines the maximum interrupt generation rate in static mode. It is similar to the *InterruptThrottleRate* of the Intel GigE NIC.

5.2 Effects of Interrupt Coalescence

In the following, we explain how IC can affect active and passive measurements as well as TCP self-clocking. We also describe the IC signature on packet train dispersion and one-way delay measurements, and show how to filter out the effects of IC.

5.2.1 Capacity Estimation.

Active bandwidth estimation tools use the received dispersion (interarrival time spacing) of back-to-back packet pairs (or trains) to estimate the end-to-end capacity of a network path [73]. The basic idea is that, in the absence of cross traffic, the dispersion of a packet pair is inversely proportional to the path capacity. Note that the received dispersion is typically measured at the application or kernel, and not at the NIC. If the receiver NIC performs IC, however, the packet pair will be delivered to the kernel with a single interrupt, destroying the dispersion that the packet pair had at the network.

Figure 55 shows the cumulative dispersion of an 100-packet Ethernet MTU (1500B) train, with and without IC, at the receiver of a GigE path. The capacity estimate, as obtained without IC, is quite close to the actual capacity of the path (about 1000Mbps). In the case of IC, however, we see that the application receives 10 to 12 packets at a very high rate (unrelated to the network capacity), separated by about 125μs of idle time. Note that, if a bandwidth estimation tool would attempt to measure the path capacity from the

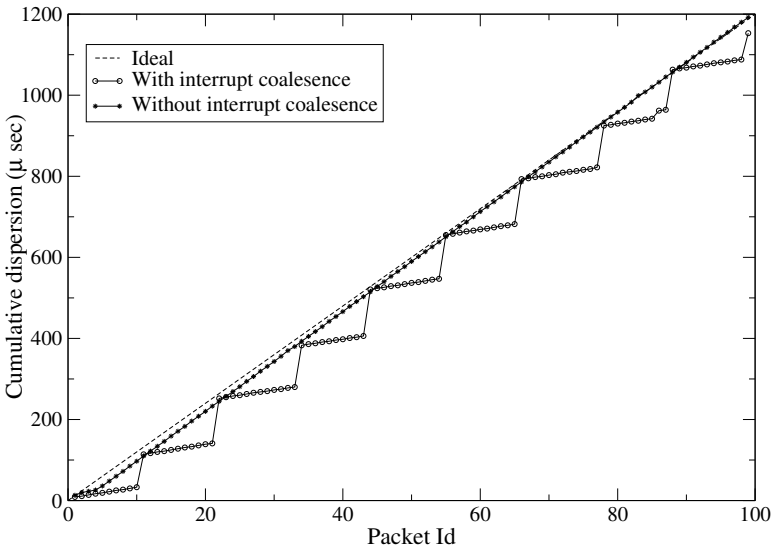


Figure 55: Cumulative dispersion in an 100-packet train with and without IC.

dispersion of packet pairs, it would fail miserably because there is not a single packet pair with the correct dispersion.

5.2.1.1 Detection of IC:

In the presence of IC, many packets will be delivered from the NIC to the kernel, and then to the application, with a single interrupt. We refer to the packets that are transferred to the application with a single interrupt as a *burst*. The dispersion of successive packets in a burst, measured at the application layer, will be equal to the latency T of the *recvfrom* system call. A measurement tool can measure T, and so it can determine which packets of a received train form a *burst*.

The presence of IC at the receiver NIC can be detected by sending a long back-to-back packet train from the sender to the receiver. If IC is enabled, the receiving application will observe a sequence of bursts of approximately the same length and duration. For example, Figure 56 shows that bursts of 10-12 packets arrive roughly every 125μs. A packet train of about 50 packets would be sufficient to detect the IC signature reliably.

An important point here is that a context switch at the receiver can cause a somewhat similar signature with IC. Figure 56 illustrates that context switching can also force several consecutive packets to be delivered to the application with the *recvfrom* period. However,

as shown in Figure 56, there is a clear difference between the signatures of context switching and IC. The former is a probabilistic event that only affects a subset of successive packets, starting from a random initial location in the packet train. The latter is a deterministic event that affects all the packets of a packet train, starting from the first, and creating a regular pattern of burst lengths and durations.

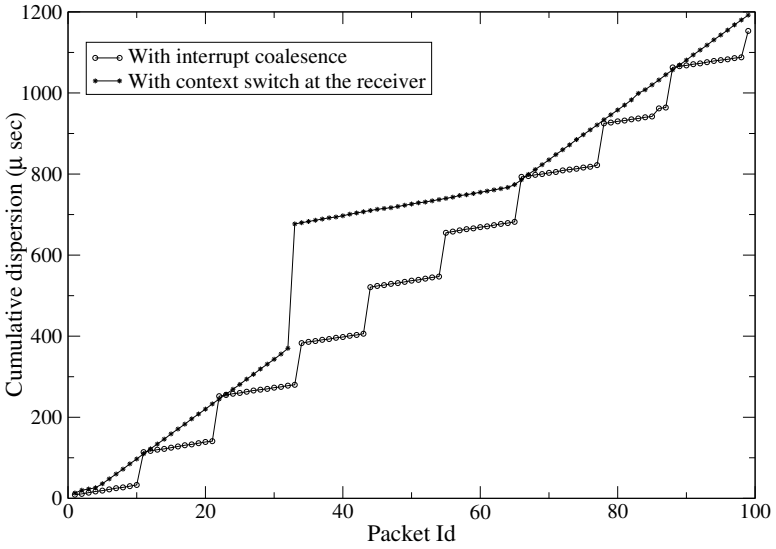


Figure 56: The signatures of context switching and interrupt coalescence in the cumulative dispersion of a packet train.

5.2.1.2 Estimation with IC:

Our capacity estimation tool, pathrate [2], uses the dispersion of consecutive packets in 50-packet trains to detect the presence of IC. To estimate the path capacity in the presence of IC, pathrate relies on the interarrivals between successive bursts. Without significant cross-traffic, the dispersion between the first packet of two consecutive bursts is equal to BL/C, where B is the length of the first burst, L is the packet size, and C is the path capacity. The train length should be sufficiently long so that at least two bursts are observed in each received train. Pathrate sends multiple packet trains and finally reports the median of the resulting capacity estimates.

5.2.2 Available Bandwidth Estimation.

Available bandwidth estimation tools examine the relationship between the *send rate* R_s and *receive rate* R_r of periodic packet streams in order to measure the end-to-end available bandwidth A [73, 59, 67, 25]. The main idea is that when $R_s > A$, then $R_r < R_s$; otherwise, $R_r = R_s$. Furthermore, when $R_s > A$, the queue of the tight link builds up, and so packet i experiences a larger queuing delay than packet $i-1$. Consequently, when $R_s > A$, the One-Way Delay (OWD) of successive probing packets is expected to show an increasing trend. On the other hand, if $R_s < A$, the OWDs of probing packets will be roughly equal. Pathload [33] and pathchirp [78] estimate the available bandwidth of a path by analyzing OWD variation

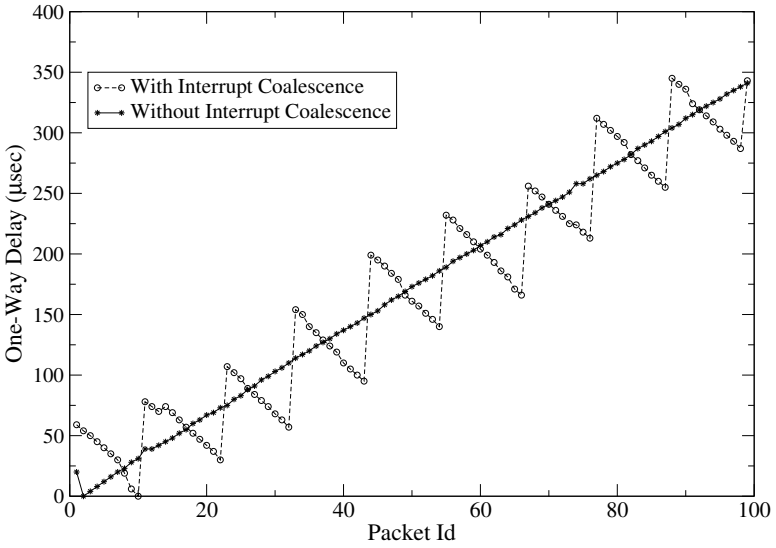


Figure 57: OWDs in 100-packet train with and without IC.

Figure 57 shows the OWDs of a periodic 100-packet Ethernet MTU stream with and without IC. The sender timestamps and transmits packets at 1.5Gbps. The available bandwidth in the path is about 940 Mbps. The OWDs of successive packets, without IC, are clearly increasing showing that the probing rate is greater than the available bandwidth. However, in the presence of IC, packets are buffered at the NIC until the interrupt timer expires. Then, the packets are delivered back-to-back to the kernel, and then to the application. We refer to all successive packets delivered with a single interrupt as a *burst*. Among the packets of a burst, the first experiences the largest queueing delay at the NIC,

while the last observes the minimum (or zero) queueing delay at the NIC. Notice that the increasing OWD trend of the packets within a burst has been destroyed due to IC.

Suppose that s_k is the send time of the k'th packet in a certain burst, while t is when the interrupt is generated. Also, let r be the latency to transfer a packet from the NIC to user space, and g be the period between successive packets at the sender. Then, the OWD of the k'th packet will be given by

$$\begin{aligned} d_k &= t + k * r - s_k \\ &= t + k * r - (s_1 + k * g) \end{aligned}$$

and so the relative OWD of packet $k+1$ will be

$$d_{k+1} - d_k = r - g \tag{37}$$

From Equation (37), we see that if $g > r$, the successive OWDs in the same burst are decreasing. In the experiment of Figure 57, we have that $g = 8\mu$s, $r \approx 3\mu$s, and so there is a clear decreasing trend in the OWDs of each burst.

Note that tools that only use packet pairs, such as [67, 78], or short packet streams (less than about 10 packets in our GigE cards) will fail to detect IC, since none of the packet pairs, or short streams, shows a correct OWD trend. On the other hand, notice that, although the OWD trend is destroyed in short timescales, the overall increasing trend in the OWDs is still preserved. Based on this observation, we have included an algorithm in Pathload to detect IC and measure available bandwidth even in the presence of IC.

5.2.2.1 Estimation with IC:

Pathload uses the same technique as pathrate (see §5.2.1.1), to detect whether the receiver NIC performs IC. When that is the case, Pathload filters out the OWD measurements that have been affected the most from IC. As explained in the previous paragraph, the queueing delay at the NIC is minimum (or zero) for the last packet of a burst. So, Pathload rejects all OWD measurements *except the last of each burst*. From the remaining measurements, we can then examine whether an increasing OWD trend exists or not.

5.2.3 Passive measurements

Passive monitors are often used to collect traces of packet headers and interarrivals. Several studies have focused on the traffic statistical characteristics (burstiness) in short timescales, by analyzing such packet traces. However, if the traces have been collected with commodity NICs that perform IC, then the packet interarrivals can be significantly altered. First, IC can make several packets appear as if they form a burst, even though that may not be the case. Second, IC can affect the correlation structure of packet interarrivals in short timescales.

We emphasize that special-purpose passive monitors typically timestamp each packet at the NIC, and so they avoid the negative effects of IC [19].

5.2.4 TCP self-clocking

Another negative effect of IC is that it can break TCP self-clocking [30]. Specifically, the TCP sender establishes its *self-clock*, i.e. determines how often it should be sending packets based on the dispersion of the received ACKs. To preserve the dispersion of ACKs at the sender, or the dispersion of data segments at the receiver, packets must be delivered to TCP as soon as they arrive at the NIC. IC, however, results in bursty delivery of data segments to the receiver, destroying the dispersion that those packets had in the network. The bursty arrival of segments at the receiver causes bursty transmission of ACKs, and subsequently bursty transmission of more data segments from the sender. The problem with such bursts is that TCP can experience significant losses in under-buffered network paths, even if the corresponding links are not saturated [92].

Figure 58 shows the CDF of ACK interarrivals in a 10-second TCP connection over a GigE path. Without IC, most ACKs arrive approximately every 24μs, corresponding to the transmission time of two MTU packets at a GigE interface. This is because TCP uses delayed-ACKs, acknowledging every second packet. With IC, however, approximately 65% of the ACKs arrive with an erratic dispersion of less than 1μs, as they are delivered to the kernel with a single interrupt. These "batched" ACKs trigger the transmission of long bursts of data packets from the sender. Note that the rest of the ACKs, even though

non-batched, they still have a dispersion that does not correspond to the true capacity of the path, misleading the TCP sender about the actual rate that this path can sustain.

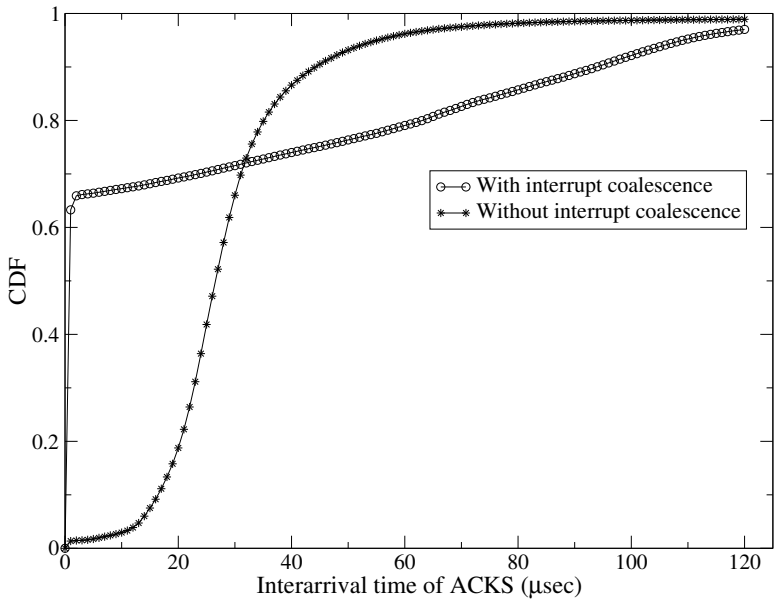


Figure 58: CDF of ACK interarrivals in a 10-second TCP connection over a GigE path.

5.3 Summary

IC is used to reduce the interrupt processing overhead and becomes necessary when the minimum packet interarrival latency becomes comparable to the per-packet interrupt processing overhead. Unfortunately, IC can affect active network measurement tools that use closely spaced packet pairs, trains, or streams. In particular, capacity and available bandwidth estimation techniques can provide incorrect results if they ignore IC. On the positive side, we developed a simple algorithm to detect the presence of IC, and to filter out the erroneous dispersion or one-way delay measurements caused by IC.

CHAPTER VI

APPLICATION OF AVAIL-BW ESTIMATION: VIDEO STREAMING

6.1 Introduction

As the "last mile" access capacity continues to grow, IP video streaming becomes more popular among users and content providers. Many experts believe that IPTV is the next "killer-application" in the Internet [15]. However, supporting video streaming and IPTV presents significant challenges. IP networks often suffer from several network impairments, including packet losses, significant jitter and one-way delays, as well as outages of unpredictable duration. Additionally, most IP networks today do not offer deterministic or statistical QoS guarantees.

Since the early nineties, several approaches for adaptive video streaming applications have been proposed. One approach is to adjust the encoding scheme and/or video frame rate in response to changes in the network state [38, 42, 76, 8]. The main drawback of such schemes, however, is that the perceived video quality varies with time, causing user dissatisfaction. Another class of approaches is to use proactive error correction techniques, such as Reed-Solomon FEC codes, or to retransmit lost packets through standard ARQ schemes [75, 47, 39]. The major drawback of FEC schemes is that they introduce bandwidth overhead even when the network does not drop packets. The drawback of retransmissions is that they require a playback delay of a few round-trip times, additional state/buffering at the sender, and reverse-path traffic (e.g., negative ACKs). Another approach is to mask the effect of lost or discarded (i.e., late) packets through the use of codec-specific error concealment techniques [79, 51]. The effectiveness of such techniques is limited however.

Even though IP networks typically use a single path from one host to another, the recent popularity of multihoming and overlay networks allows content providers to choose between several network paths towards a given receiver [94]. Such path diversity gives video streaming one more adaptation option: to dynamically switch from one path to

another depending on the observed (or predicted) performance in the candidate paths. This technology uses network-level measurements, such as loss or jitter, and it has been shown that it can quickly react to congested paths or outages [86, 5]. A variation of this approach is to combine path diversity with Multi-Description Coding (MDC) techniques [9, 7, 93], and use multiple paths simultaneously. The studies in this area rely on loss rate, delay or TCP throughput measurements, and they typically perform these measurements using "dummy" probing packets.

In this work, we consider an overlay-based video streaming architecture in which the objective is to maximize the perceived video quality through dynamic overlay path selection. A novel aspect of our study is that the network measurements that drive the path selection process rely on available bandwidth (avail-bw) estimation. The avail-bw of a network path is defined as the residual capacity at the path's bottleneck, and so it represents the maximum additional load that the path can carry before it becomes saturated [33]. The reason we focus on avail-bw is because this metric can determine whether a path has enough capacity to carry a video stream *before* we switch the stream to that path. Other network-layer metrics, such as jitter or packet loss rate, can only determine whether a path is *already* congested, causing degradation in the video quality at the receiver [84, 85, 12]. We also show how to modify an existing avail-bw estimation technique, described in [33], so that the measurements are performed using application packets, rather than "dummy" probing packets, eliminating the measurement overhead in the currently selected path. We evaluate the video quality based on the VQM technique described in the ITU-T recommendation J.144 [1]. It has been shown that VQM is superior to other video quality metrics, such as PSNR, because the VQM score is more representative of the user-perceived video quality [56]. With a series of repeatable experiments in a controlled environment we compare the VQM score of path selection schemes based on jitter, loss rate, and various percentiles of the avail-bw distribution. The main result of this experimental study is that performing path selection based on *the estimated lower bound of the avail-bw variation range* performs significantly better in terms of VQM score, path switching frequency, and probability of aborting an ongoing video stream.

The rest of the chapter is organized as follows. §6.2 presents an overlay-based video streaming architecture. §6.3 describes the path selection techniques that we evaluate. The experimental methodology is described in §6.4, and the results are presented in §6.5. §6.6 gives a brief comparison between adaptive path selection and a simple (but commonly used) FEC scheme. We conclude in §6.7.

6.2 VDN architecture and in-band measurements

In this section, we present the high-level architecture of an overlay-based video streaming architecture, referred to as the *Video Distribution Network (VDN)*. A content provider constructs a VDN by deploying several overlay nodes that will act as either overlay ingress/egress nodes or as intermediate nodes (see Figure 59). Each VDN node runs a *Measurement Module (M-module)* to measure the performance (jitter, loss rate, avail-bw) of the overlay links from that VDN node to its neighbors. The VDN also runs a link-state protocol so that each node is aware of the latest state in all VDN links. We only consider VDN paths with at most one intermediate overlay node, based on the results of [94]; additional intermediate nodes are rarely needed in practice. The path of a video stream is determined at the ingress VDN node, i.e., we use source routing. The egress VDN node removes any VDN headers and delivers the stream to the receiver.

For the purposes of this chapter, the most important aspect of the VDN architecture is the M-module. In our implementation, the M-Module relies on active measurement to estimate the packet loss rate, jitter, and the avail-bw variation range at a given overlay link. The loss rate is measured as the fraction of lost packets in a 1-Mbps stream of 1500-byte packets. The jitter is estimated as the maximum absolute difference between the spacing of consecutive packets at the receiver relative to their spacing at the sender, using the same probing stream. The avail-bw variation range is measured as described in chapter 3 and chapter 4.

One feature of the M-Module is that it uses the application's video packets to perform *in-band* network measurement. The in-band approach eliminates the measurement overhead, at least in the path that is currently selected for video streaming (we still use out-band

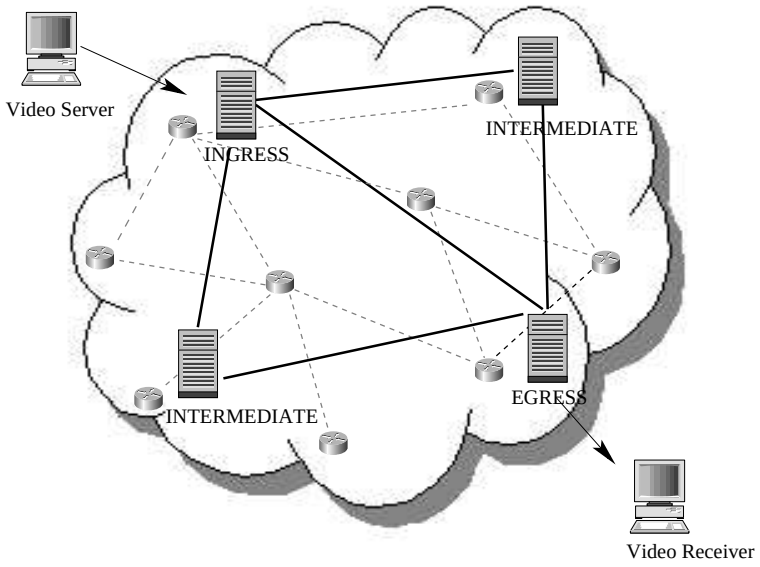


Figure 59: VDN architecture.

measurements with empty probing packets in paths that do not transfer any video streams). Using video packets for the estimation of loss rate or jitter is relatively straightforward. The estimation of avail-bw, on the other hand, requires shaping the video stream at a different rate than the transmission rate at the sender. In the following paragraph we describe how the M-module shapes the video stream to a particular rate at the input of each overlay link.

Suppose that N video packets of size L arrive at a VDN node at a rate R_v. The arrival time of the i^{th} packet is t_i^a. The objective of the local M-module is to shape those N packets at a probing rate R_p, so that it can measure whether R_p is smaller or larger than the avail-bw in the outgoing overlay link. To do so, the N incoming packets are delayed so that their output spacing is L/R_p. If t_i^d is the departure time of the i^{th} packet, then $t_i^d - t_{i-1}^d = L/R_p$ and $t_i^d = t_i^a + \delta_i$, where δ_i is the delay introduced to the i^{th} packet. For instance, if $R_p > R_v$, the M-module shapes the packet stream by introducing a decreasing amount of delay in successive packets; otherwise, the M-module introduces an increasing amount of delay in successive packets (see Figure 60). The M-module adds the value δ_i into the VDN header of packet i. In some cases, the receiving application may demand that the video stream arrives at the rate R_v in which it was sent to (e.g., for clock synchronization). In that case, the egress VDN node can reshape the video stream rate back to the initial

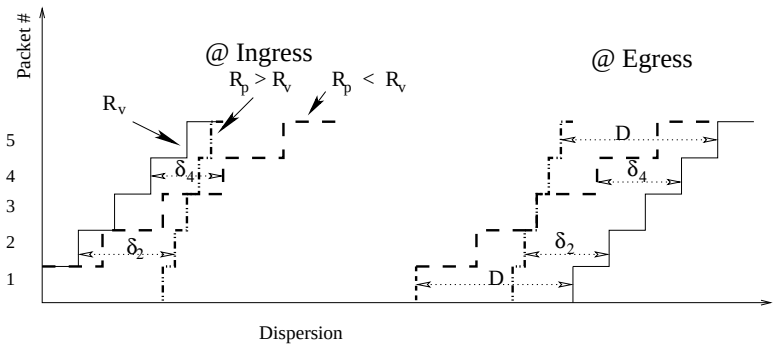


Figure 60: Shaping of video packets to a probing rate R_p for avail-bw estimation.

transmission rate at the sender, by delaying each packet by $D-\delta_i$, where D is the maximum cumulative delay that can be introduced to the packet from the M-modules at the ingress and intermediate nodes and δ_i is the corresponding cumulative delay that the i^{th} packet experienced.

6.3 Path selection schemes

In this section, we describe the four path selection schemes we evaluate. The schemes are distinguished based on the choice of the key measured network performance metric.

Loss based path selection (LPS): In LPS, we monitor the average loss rate in all candidate paths during 3-second periods. The path with the minimum loss rate is selected. If the currently used path has zero loss rate, then we do not switch to another path even if there are other loss-free paths.

Jitter based path selection (JPS): The jitter of successive packet pairs is also measured over 3-second periods. The path with the minimum 90^{th} percentile of jitter measurements is selected. If the minimum jitter is practically the same in more than one paths, then JPS selects the path with the lowest loss rate. If the loss rate is also equal, then JPS stays at the current path if that is one of the best paths, or it randomly picks one of the best paths otherwise.

Avail-bw based path selection (APS): This scheme has two variations. In the first, we use the average avail-bw (A-APS). In the second, we use the lower bound of the avail-bw

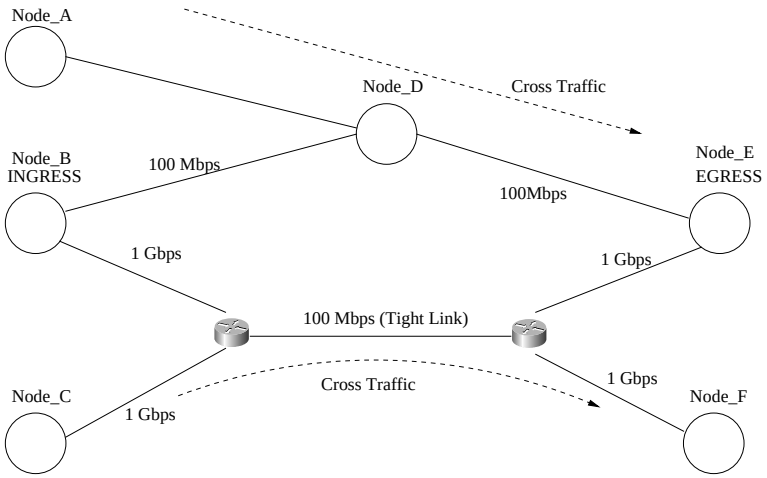


Figure 61: Testbed.

variation range (L-APS) (see 4 for more details). A new avail-bw estimate results in almost every 3 seconds, similar to LPS and JPS. If the avail-bw estimate (average or lower bound) in the currently selected path is greater than twice the video transmission rate, then we stay in that path. Otherwise, we choose the path with the highest avail-bw estimate; note that this may still be the currently selected path.

6.4 Experimental setup

We have evaluated the performance of the previous path selection schemes with controlled experiments in the testbed of Figure 61. The video stream is transmitted from node B to E through either the direct path, or through the path that traverses node D. The video stream is a 2-minute clip formed by combining SMPTE test video sequences [20], namely *flower garden, susie, birches and football*, and encoded in MPEG-2 with average rate 6Mbps. The VLC player [88] transmits the stream to the network. The stream is initially routed through the direct path. The M-modules run at nodes B, D, and E, and they perform both network measurement and the path selection process. Both paths carry cross traffic and they are occasionally congested. The cross traffic in each path is generated by replaying NLANR packet traces, collected from various university access links [63]. The average cross traffic rate is set to the desired value by scaling the packet interarrivals by the appropriate factor.

We compare the performance of various path selection algorithms based on three criteria: *video quality*, *user-abort probability* and *path switching frequency*. The video quality is measured using the VQM tool [90], which implements the ITU-T J.144 recommendation [1]. VQM compares the original video stream with the received video stream, and it reports a metric between 0 and 1. Note that *a lower VQM scores correspond to better video quality*. It has been shown that the VQM score correlates very well with the user-perceived video quality (MOS score). The VQM software supports five models to evaluate video quality, described in detail in the NTIA Handbook [71]. In this work, we use the *television model*. In the following graphs we report the minimum, average, and maximum VQM score from the five runs of each experiment. The five runs differ in terms of the initial phase between the video clip and the cross traffic traces.

The *user-abort probability* focuses on the short-term variations of the VQM score. The idea is that if the VQM score is too high (poor quality) during a time window, then the user would either abort the video stream or she would be unsatisfied. We measure the VQM score of consecutive 10-second video segments. A video stream is considered *aborted* if one of the following two conditions is met: either the VQM score in a single segment is larger than 0.55, or two consecutive video segments have VQM scores larger than 0.35. We chose these values based on extensive subjective tests of several video streams under different conditions. To estimate the user-abort probability, we measured the fraction of aborted video stream in 30 experiments.

The last evaluation metric is the total number of path switching events. Even though the path switching frequency does not affect the video quality, it is an important aspect of any dynamic routing mechanism from the network operator's perspective. Frequent path switching of large traffic volumes can affect the network stability and traffic engineering. Consequently, even though our primary interest is to optimize video streaming quality, we would also like to avoid unnecessary path switching.

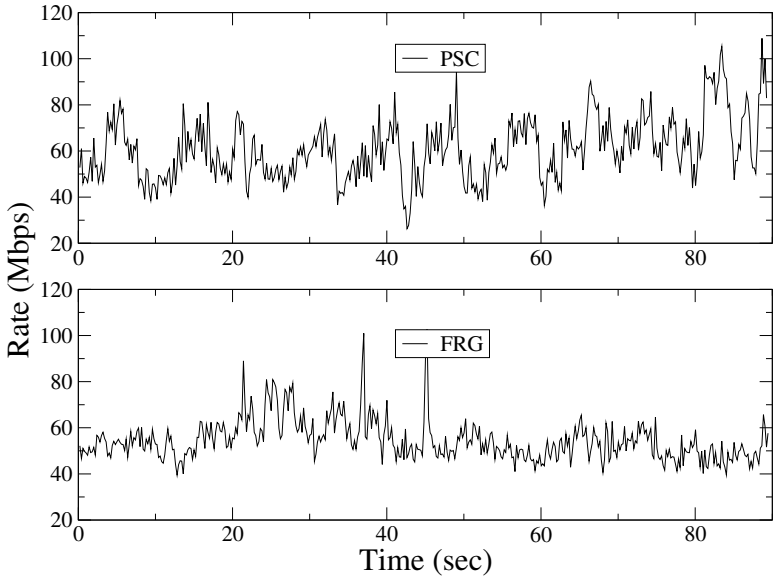


Figure 62: Two NLANR cross traffic traces.

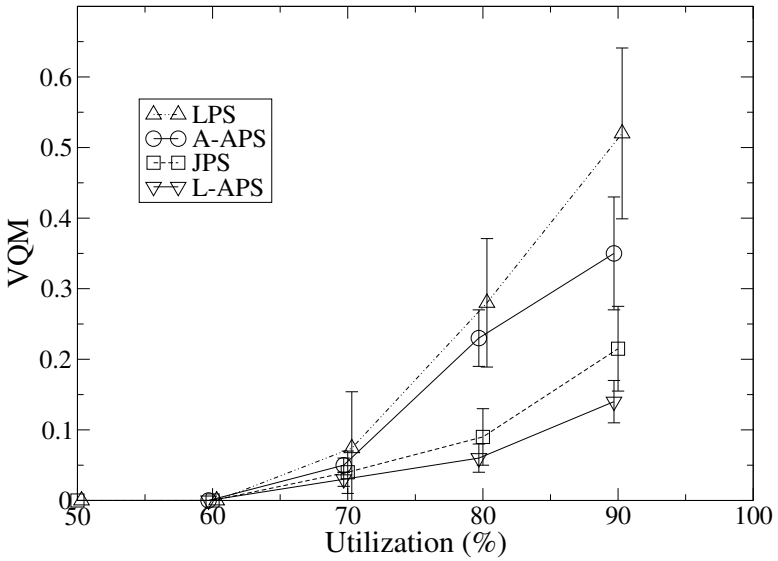


Figure 63: VQM scores for the four path selection schemes.

6.5 Results

Figures 63, 64 and 65 show the performance of the considered path selection schemes under different load conditions (i.e., utilization of the bottleneck link in each path). The PSC trace shown in Figure 62 is replayed at the direct path, while the FRG trace is replayed at the indirect path. We adjust the average rate of each trace to achieve the desired utilization. Even though we set the long-term average rate of the two traces at the same value, there are time periods where one path is congested while the other is not. Note that there are time periods, mainly in higher load conditions, where both paths are congested. Obviously, path switching techniques cannot avoid congestion in that case, but they can still choose the least congested path.

Figure 63 shows the overall VQM score for each path selection scheme. LPS clearly performs poorly since it only reacts after congestion has affected the currently selected path. The A-APS scheme does not perform much better than LPS. The reason is that the average avail-bw does not capture the variability of the avail-bw distribution, and so the two paths appear as almost equally good in most of the time. The JPS and L-APS schemes have comparable performance and they are clearly better than A-APS and LPS. This is because both JPS and L-APS are able to detect the onset of queuing delays in the currently selected path, before that path becomes congested. Note that L-APS is slightly better than JPS, especially in the case of 90% utilization.

Figure 64 shows the number of path switching events in the same set of experiments. The main observation is that L-APS has the lowest path switching frequency. JPS causes significantly more path changes, and is comparable to A-APS. This is because JPS relies on a comparison of the maximum jitter in the two paths, and so a minor variation in the jitter, which can result from an short-lived cross-traffic burst, may trigger JPS to switch paths. Instead, L-APS does not switch paths if the currently selected path provides a large safety margin, in terms of avail-bw, for the given video stream rate.

Figure 65 shows the user-abort probability, i.e., the fraction of aborted video streams. The ranking of the four path selection schemes is as in the case of the long-term VQM score in Figure 63.

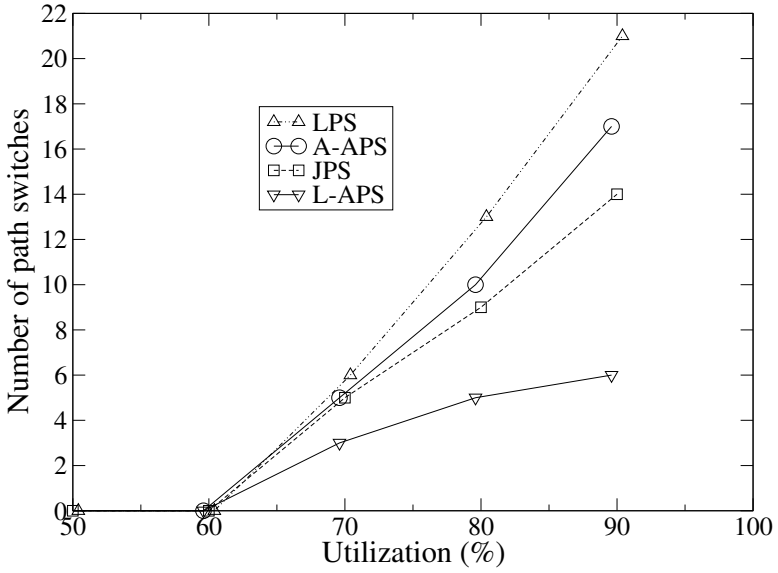


Figure 64: Path switching frequency.

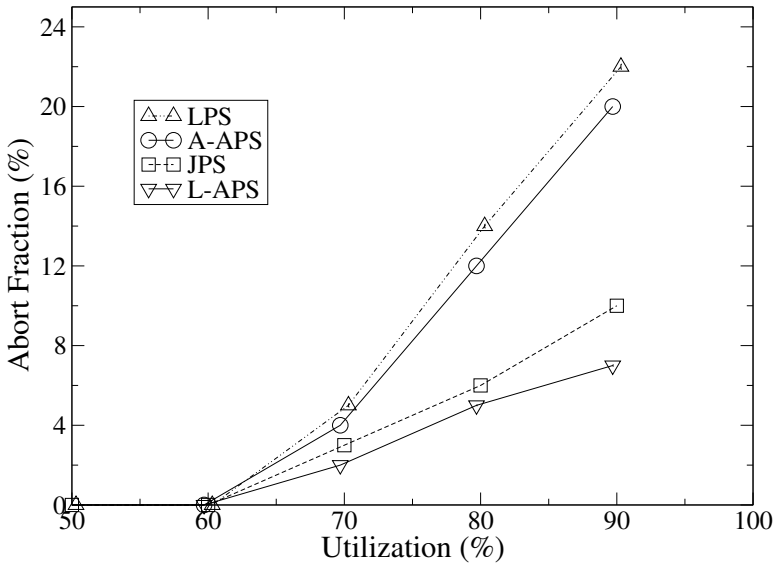


Figure 65: User-abort probability.

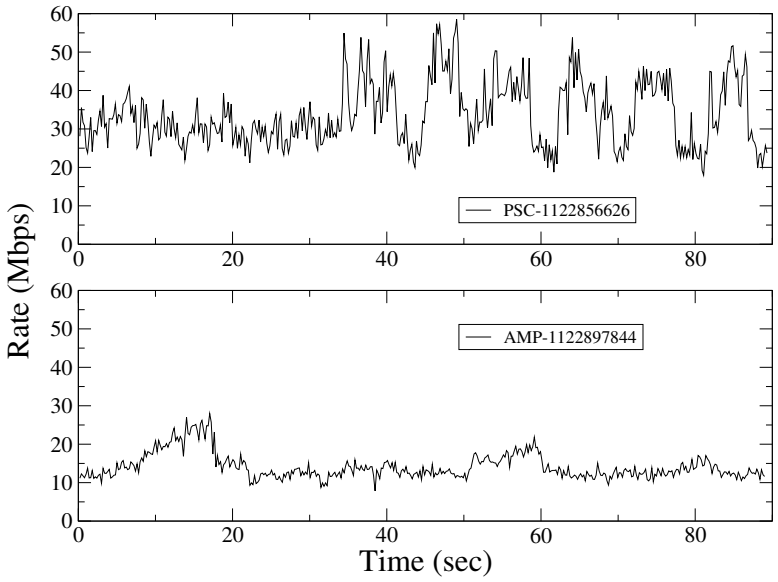


Figure 66: Cross traffic traces with instances of non-stationarity.

We next show some results for the traces shown in Figure 66. These traces include instances of traffic non-stationarity. In the PSC trace, the traffic rate varies abruptly between 20Mbps and 45Mbps during the second half (level shifts). On the other hand, the AMP trace exhibits periods of slowly increasing traffic load (notice the traffic "ramp"). We are interested to examine the effectiveness of the considered path selection schemes under such traffic conditions. In these experiments, we use the same trace in both paths. In one path, the trace playback starts from the beginning, while in the other path the trace playback starts from the middle.

Figures 67 and 68 show the VQM scores for the PSC and AMP traces, respectively. Note that, overall, the level shifts of the PSC trace cause higher VQM scores compared to the smoother AMP trace. L-APS performs clearly better in this case than the JPS scheme. Note that there is a difference in the relative performance of JPS and A-APS in the two traces. The reason is that, even though JPS is more proactive than A-APS in switching paths, in the AMP trace the A-APS scheme performs slightly better because it can detect the slowly decreasing level of avail-bw during the traffic ramp.

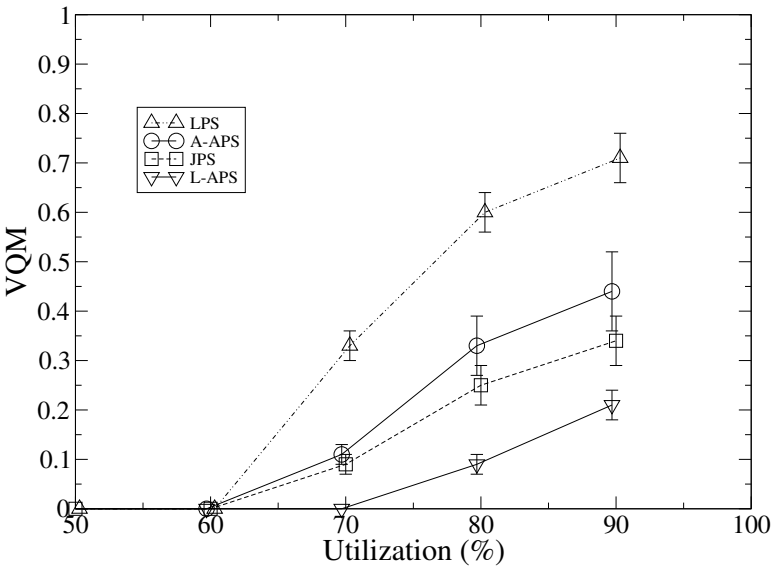


Figure 67: VQM scores for the PSC trace.

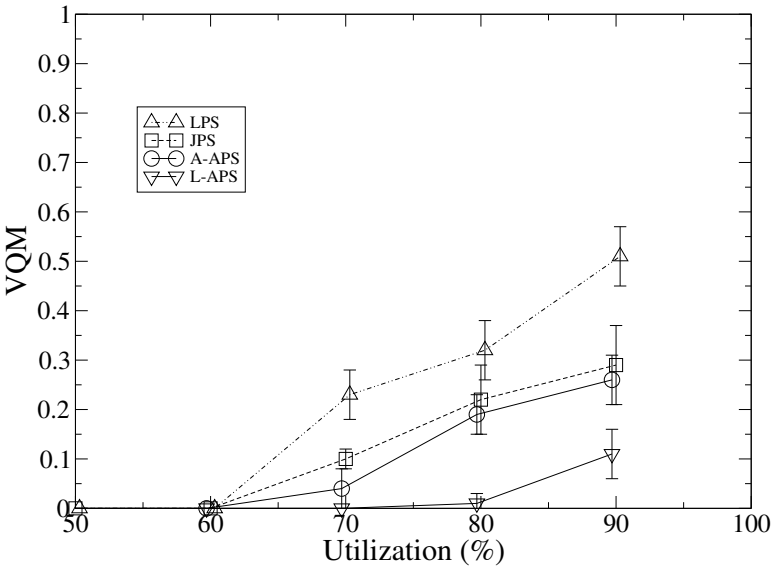


Figure 68: VQM scores for the AMP trace.

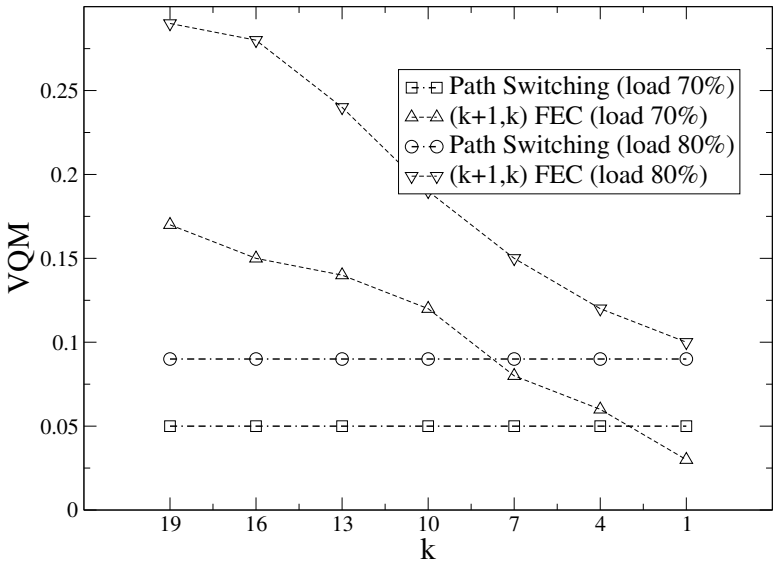


Figure 69: Path switching versus FEC.

6.6 Path switching versus FEC

In this section, we conduct a preliminary comparison between path switching techniques and FEC-based loss recovery. A commonly used FEC scheme is the Reed-Solomon (RS) code [75, 47, 39]. In an (n, k)-RS code, $n-k$ out of n packets carry FEC packets. An (n, k) RS-code can recover from all losses in a block of n packets if at least k of those n packets are received. The main drawback of FEC-based schemes is their transmission overhead $n-k/k$.

Here, we evaluate the simplest form of FEC in which $n = k + 1$. This instance of RS-coding is equivalent to sending a single parity packet after every k data packets. We compare the VQM score of this technique with L-APS for two load conditions: 70% and 80% bottleneck utilization using the NLANR traffic trace (BWY-1063315231). Figure 69 shows the results for different values of k. At the far left, k=19 corresponds to a transmission overhead of about 5%, while at the far right, k=1 corresponds to 100% overhead. Note that path switching with L-APS performs consistently better than FEC, except the case of k=1. The main reason is that path switching can often avoid congestion altogether, while FEC attempts to recover from the effects of congestion, which is not always possible. Additionally, the FEC scheme we evaluate here is not effective in dealing with the bursty nature of congestion-induced packet losses.

6.7 Summary

This work focused on the use of measurement-driven path selection techniques for video streaming. We showed that if the path selection is driven by a conservative estimate of the available bandwidth, then the resulting video streaming performance is significantly improved compared to other commonly used network metrics. An interesting open problem is to develop path switching mechanisms that are driven by direct video quality measurements at the receiver. Another open problem is to evaluate the performance of a hybrid approach using both FEC and path selection mechanisms.

CHAPTER VII

RECENT WORK IN AVAIL-BW ESTIMATION

Estimating the available bandwidth (avail-bw) of end-to-end network paths with active measurements has attracted significant interest recently. Several estimation techniques and software tools have been developed, IGI [25], Pathchirp [78], Spruce [81], and Bfind [3], while several more techniques are currently being reviewed or developed. On the positive side, this work reflects that the networking community recognizes the importance of the avail-bw metric and its many practical applications. On the negative side, however, we find that some key issues regarding the avail-bw definition, estimation, and validation remain vague or misinterpreted in both the general understanding of our community, but also in published work.

In this chapter, we review the recent work in the area of avail-bw estimation. Additionally, we clarify what we think of as the ten most important misconceptions about avail-bw estimation. Our objective is not to debunk recent work or to claim that some estimation techniques (or our estimation technique in chapter 2) are better than others. Actually, the clarification of the following issues may lead to better estimation techniques, and also, it may allow a more fair comparison between existing measurement tools.

7.1 Avail-bw estimation at a single link with fluid traffic

All the existing avail-bw estimation techniques are based, at least in terms of their basic idea, on a single-link model with fluid cross traffic of (constant) rate. Let C_t be the capacity of that link and R_c be the rate of cross traffic, with the avail-bw being $A = C_t - R_c > 0$. The fluid assumption means that during any time interval of length τ, the amount of arriving cross traffic at the link is $R_c\tau$.

Suppose that we send to this link a periodic probing stream with rate R_i and packet size L; the length of the stream does not matter at this point. The interarrival between two consecutive probing packets is $\delta_i = L/R_i$. The amount of cross traffic that arrives at

the link during any interval of δ_i is $X_c = R_c\delta_i$, and the total amount of traffic that arrives at the link in the same interval is $L + X_c$ (including one probing packet). On the other hand, the maximum amount of traffic the link can transmit in the same interval is $C_t\delta_i$. It is straightforward to show that if $R_i > A$, then the link receives more traffic than it can service, i.e. $L + X_c > C_t\delta_i$.

In that case, when $R_i > A$, the extra arriving traffic at the link accumulates at the link's buffers, increasing the queue size with every probing packet we send. It is also easy to show that the queue size increase Δq during an interval of length δ_i is

$$\Delta q = \delta_i(R_i - A) = L\frac{R_i - A}{R_i} \qquad \text{if} \quad R_i > A \tag{38}$$

Two ways to detect that $R_i > A$ from end-to-end measurements are through increase in the One-Way Delay (OWD) of the probing packets, and through a lower output rate R_o at the receiver compared to the input rate R_i at the sender. Specifically, the OWD increase Δd between two successive probing packets during an interval of length δ_i is determined by the queue size increase in the same interval

$$\Delta d = \frac{\Delta q}{C_t} = \frac{L}{C_t}\frac{R_i - A}{R_i} \qquad \text{if} \quad R_i > A \tag{39}$$

Also, the interarrival between two consecutive probing packets after the link (at the receiver) is $\delta_o = \delta_i + \Delta d$, and so the rate with which the probing stream arrives at the receiver is

$$R_o = \frac{L}{\delta_o} = \frac{R_i C_t}{C_t + (R_i - A)} \qquad \text{if} \quad R_i > A \tag{40}$$

The key point here is that the output rate is less than the input rate $R_o < R_i$ when $R_i > A$.

On the other hand, if $R_i \leq A$ the link can transmit all the received traffic and so $\Delta q = \Delta d$=0, meaning that the OWDs do not increase, and $R_o = R_i$ meaning that the output rate is equal to the input rate.

We next identify two different avail-bw estimation approaches that are based on the previous model. All the existing estimation techniques and tools are based on these general approaches.

7.1.1 Direct probing

In direct probing, each probing stream results in a sample of the avail-bw. The sender transmits a periodic probing stream of rate R_i and receiver measures the output rate R_o. The basic idea is that, if R_i is larger than the avail-bw, Equation (40) can be solved for the only unknown A

$$A = C_t - R_i(\frac{C_t}{R_o} - 1) \tag{41}$$

Note that direct probing samples the avail-bw process with each packet train, as long as the input rate is sufficiently high. The main assumption in the direct probing approach, however, is that the tight link capacity C_t is known. We will return to this point in §7.3.

7.1.2 Iterative probing

In iterative probing, we do not need to know the capacity of the tight link. The sender transmits a periodic probing stream k with rate $R_i(k)$. The rate $R_i(k)$ varies either linearly, or as a function of the outcome of previous streams. If the k'th stream gives $R_o(k) < R_i(k)$, or if the OWDs of that stream are increasing, then we know that $R_i(k) > A$; otherwise, it is $R_i(k) \leq A$. The basic idea is that, through a sequence of streams with different rates, iterative probing can converge to A. Thus, the basic equation behind iterative probing is:

$$R_o(k) < R_i(k) \quad \text{if } R_i(k) > A; \qquad \text{else } R_i(k) \leq A \tag{42}$$

A key point about iterative probing is that it does not sample the avail-bw process; instead, it only samples whether a rate is larger than the avail-bw or not.

7.2 Classification

We next briefly review and classify each existing avail-bw estimation technique. The techniques are presented in the order in which they were published.

Delphi: Delphi is the canonical example of direct probing [77]. An interesting point about that work is that it assumes a multifractal model for the path's cross traffic. Similar to other direct probing techniques, Delphi assumes that the avail-bw is limited by a single tight link and that the capacity of the tight link is known or it can be estimated.

Trains of packet pairs (TOPP): TOPP is the canonical example of iterative probing [59, 60]. A probing stream at a rate R_i consists of several packet pairs with that rate. The rate increases linearly in successive streams. TOPP estimates the avail-bw as the maximum input rate R_i that is not larger than R_o. An interesting point about TOPP is that it attempts to also estimate the capacity of the tight link, as well as the avail-bw and capacities of secondary bottlenecks in the path.

Pathload: Pathload is an estimation tool (described in chapter 3) that we developed, and it is based on iterative probing. A difference with TOPP is that Pathload varies the probing rate in a "binary search" manner, rather than linearly. Pathload was the first tool to consider the variability of the avail-bw process, and this is why it estimates a *variation range* rather than a single estimate. Furthermore, it infers the relation between the input rate and the avail-bw based on statistical analysis of the OWD trends, rather than based on the ratio R_o/R_i. Also, in [33] we considered the effect of the probing stream length on the variation range of the avail-bw, and explained the differences between TCP throughput and avail-bw. We will return to the previous points in §7.3.

Pathchirp and S-chirp: Pathchirp is based on iterative probing [78]. Instead of sending periodic packet streams, Pathchirp sends "chirps", i.e., N packets with exponentially increasing interarrivals between successive packets. Therefore, the packet pairs of a single chirp probe the network at a wide range of rates, while a single chirp of length N probes $N-1$ different rates. Note that Pathchirp's efficiency in terms of the number of probing rates per packet is due to the use of consecutive packet pairs. Pathchirp reports a single estimate of the avail-bw during a sequence of chirps.

A similar approach, referred to as *Smoothed-chirp* (S-chirp), was proposed in [67] and it is also based on iterative probing.

IGI/PTR: Hu and Steenkiste proposed two estimation techniques (IGI and PTR) in [25]. PTR is quite similar with TOPP, in the sense that it is also based on iterative probing, performs linear rate variation, and reports a single estimate. A difference with TOPP is

that it uses packet trains of 60 packets rather than packet pairs.

IGI is harder to classify. It uses an equation that is similar to (41), and so it can be viewed as direct probing. The input rate R_i, however, is chosen in an iterative manner, as in TOPP and PTR. It is not clear what is the benefit of also using the direct probing equation, given that at the end of the iterative phase IGI already has an estimate of A. IGI also uses packet trains of 60 packets. The capacity of the tight link is estimated using *bprobe*, which is an end-to-end capacity estimation tool [14].

Spruce: Spruce is also based on direct probing [81]. Spruce uses 100 packet pairs, instead of packet trains, to collect its avail-bw samples. The input rate is chosen to be roughly equal to the tight link capacity, which is assumed to be known. To emulate *Poisson sampling*, different packet pairs are spaced with exponential interarrivals.

Bfind: BFind is also based on iterative probing [3]. It differs from previous tools in that it does not require control at both ends of the path but it is based on ICMP TTL-expired responses from the intermediate routers in the path. Specifically, the sender probes the end-to-end path with UDP streams of gradually increasing rate, while monitoring at the same time the RTTs to successive routers with repeated traceroute measurements. An increase in the RTT to a particular link reveals an increasing queue size, which implies that the corresponding input rate is larger than the avail-bw at that link.

In addition to the above tools and techniques, several new tools have appeared in the literature in last couple of years. Antoniades et al. presented a single-ended avail-bw estimation tool, called abget [6], which uses only TCP packets coupled with iterative algorithm similar to pathload. Hu et al. proposed a tight link detection tool called pathneck [24], which is based on direct probing approach. The key assumption in the tool is that the packet train length will increase only at the link where the avail-bw is less than train rate. Authors used similar approach in [26].

7.3 Fallacies and pitfalls [1]

7.3.1 Pitfall: Ignoring the variability of the avail-bw process.

Suppose that we collect k independent samples of the avail-bw process $A^\tau(t)$. Even if we could measure each sample with perfect accuracy, the variability of $A^\tau(t)$ means that the sample mean $m_A(k)$ will be different than the population mean μ_A. The variance of $m_A(k)$ is

$$\mathrm{Var}[m_A(k)] = \frac{\mathrm{Var}[A^\tau]}{k} \tag{43}$$

where the population variance $\mathrm{Var}[A^\tau]$ of the avail-bw process depends on the averaging time scale as discussed in §7.1. The fact that the variance of the sample mean decreases with the number of samples, or the fact that the population variance decreases with the time scale τ are basic statistical facts. Unfortunately, the avail-bw estimation literature includes accuracy comparisons between estimation techniques that use a different number of samples, or between estimation techniques that use different averaging time scales.

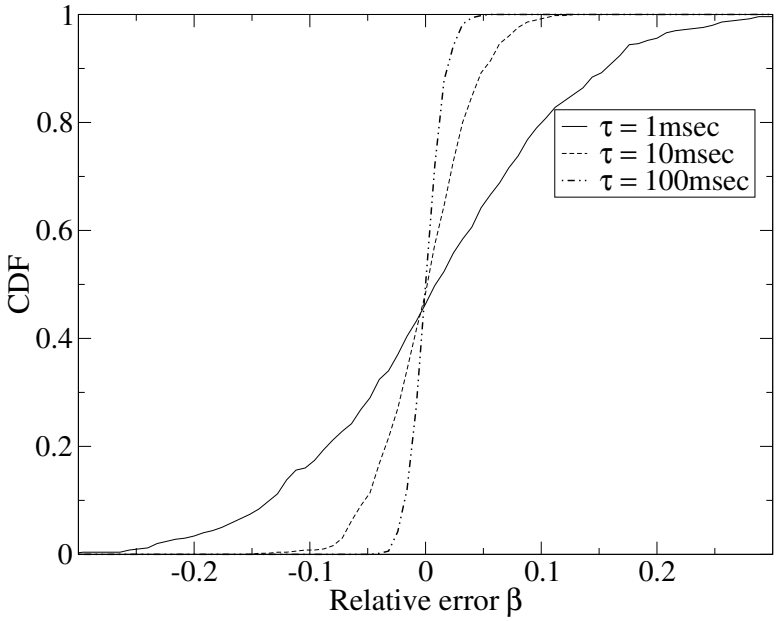


Figure 70: Relative error β of the sample mean m_A for three averaging time scales.

To illustrate the magnitude of the errors that can be caused simply due to sampling,

[1] We adopt the heading "Fallacies and Pitfalls" from the well-known Computer Architecture textbook of Hennessy and Patterson.

even with perfect measurement of each sample, we analyze an NLANR packet trace from an internet link[2]. The trace provides us with accurate knowledge of the avail-bw process in a wide range of time scales τ. Our experiment consists of collecting sets of k=20 avail-bw samples from the trace, using Poisson sampling, calculating the sample mean m_A, and then computing the relative error $\beta = \frac{m_a - \mu_A}{\mu_A}$, where μ_A is the mean avail-bw in the trace. Figure 70 shows the CDF of β for three averaging time scales. Note that unless if τ is 10ms or more, significant errors should be expected with 20 samples, simply due to the variability of the avail-bw process. Especially in short time scales, say 1ms, the number of samples that is required for a reasonably low β (say $< 5\%$) can be in hundreds.

7.3.2 Pitfall: Ignoring the relation between probing stream duration and averaging time scale.

The averaging time scale τ is an important parameter in avail-bw estimation, and we should expect that different applications will be interested in different values of τ. With the exception of [33], the intimate relation between τ and the probing stream duration has been ignored.

It is important to understand that we can control the averaging time scale with the probing stream duration, since the latter determines the length of the time interval in which we interact with the cross traffic at the tight link. To illustrate, we performed a single-hop simulation of direct probing. The capacity of the link is 50Mbps, the average avail-bw is 25Mbps, the cross traffic is Poisson, and the input probing rate is 40Mbps. An avail-bw estimate is obtained using direct probing with each stream. The simulations are repeated five times, for different values of the stream duration (25ms, 50ms, 100ms, 150ms, and 200ms). We then compare the standard deviation of 100 avail-bw samples with the population standard deviation (derived from the simulation packet trace) for the corresponding averaging time scale. Figure 71 shows that the population and sample standard deviations are almost equal. In summary, the probing duration should not be viewed as an "implementation

[2]Trace ANL-1070432720 from the OC-3 access link of Argonne National Laboratory. NLANR PMA project is supported by the NSF (ANI-9807479 and ANI-0129677) and by the National Laboratory for Applied Network Research.

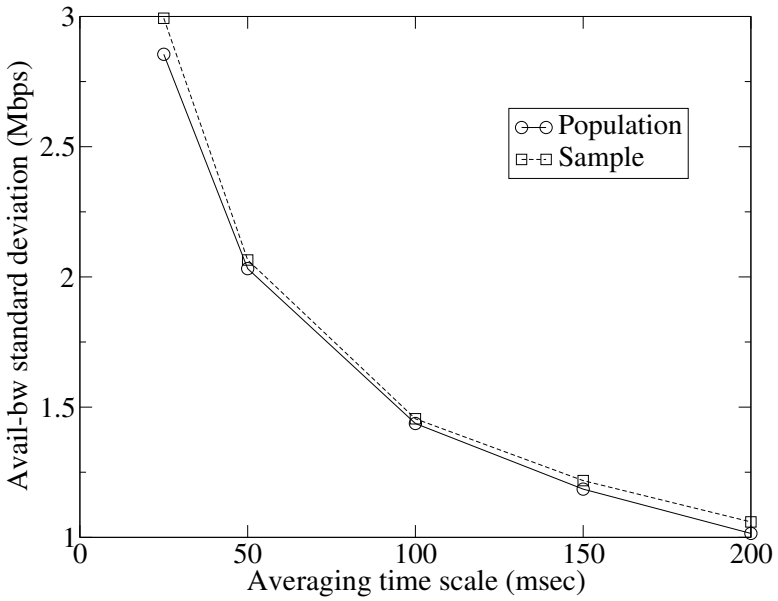


Figure 71: The probing stream duration controls the averaging time scale τ.

parameter", but as the knob that controls the averaging time scale.

7.3.3 Fallacy: Faster estimation is always better.

Some avail-bw estimation tools have been proposed on the premise that they are faster than other tools. Using fewer streams or shorter streams, however, reduces the estimation latency with a penalty in terms of accuracy. For instance, decreasing the probing stream duration decreases the averaging time scale. Since the variance of the avail-bw process increases with a smaller averaging time scale, it is clear from (43) that the variance of the avail-bw sample mean would be increased (for the same number of samples). In general, the stream duration and the number of streams should be inputs to an estimation technique, as knobs that control the estimation accuracy and the measurement overhead/intrusiveness. Additionally, comparisons between different techniques should take into account the tradeoff between estimation latency and accuracy.

7.3.4 Fallacy: Packet pairs are as good as packet trains.

It is true that with fluid cross traffic there is no difference between using packet pairs or longer trains. In practice, however, the cross traffic consists of packets with discrete packet

sizes, commonly following a strongly modal distribution. Consequently, the amount of cross traffic that interferes between each probing packet pair can often take just some discrete values, such as one 1500B packet, two 40B packets, etc. For a given number of samples, and for a given avail-bw process, the avail-bw estimation error will be higher when the cross traffic consists of a few large packets rather than many small packets. In the extreme, if we could reduce the cross traffic packet size L_c to almost zero, the traffic would behave as a fluid model.

Table 2: Effect of cross traffic packet size L_c on the relative error β for four sample sizes k.

	k=10	k=20	k=50	k=100
L_c=40B	≈0	≈0	≈0	≈0
L_c=512B	31%	8%	5%	2.5%
L_c=1500B	40%	20%	8%	2%

To illustrate this point, we performed single-hop simulations (with the same topology and parameters as in the previous paragraph) with three different values of L_c, while keeping the average avail-bw constant. The probing packet size is L=1500B. Table 2 shows the relative error β for four sample sizes. Even though packet pairs are able to provide good accuracy when the cross traffic consists of small packets (40B), the presence of larger packets makes packet pairs significantly inaccurate.

7.3.5 Pitfall: Estimating the tight link capacity with end-to-end capacity estimation tools.

As previously mentioned, direct probing techniques require the knowledge of the tight link capacity C_t. It is often assumed that C_t can be estimated with end-to-end capacity estimation tools. The latter, however, would estimate the capacity C_n of the narrow link, which may be less than C_t. This is often the case, for instance, when C_n is limited by a Fast Ethernet interface while the tight link is an OC-3 or OC-12 link that has less avail-bw than the narrow link.

7.3.6 Fallacy: Increasing One-Way Delays is equivalent to $R_o < R_i$.

In the fluid model, increasing OWDs is equivalent to $R_o < R_i$; see Equations (39) and (40). With real cross traffic, however, the time series of OWD measurements carries much more information than the ratio of the average input and output rates. The reason is rather obvious: the OWDs are many values, and so they can be analyzed with statistical tools to detect trends, measurement noise, level shifts, etc. The ratio R_o/R_i, on the other hand, is a single number, hiding much information about what happened during that probing stream.

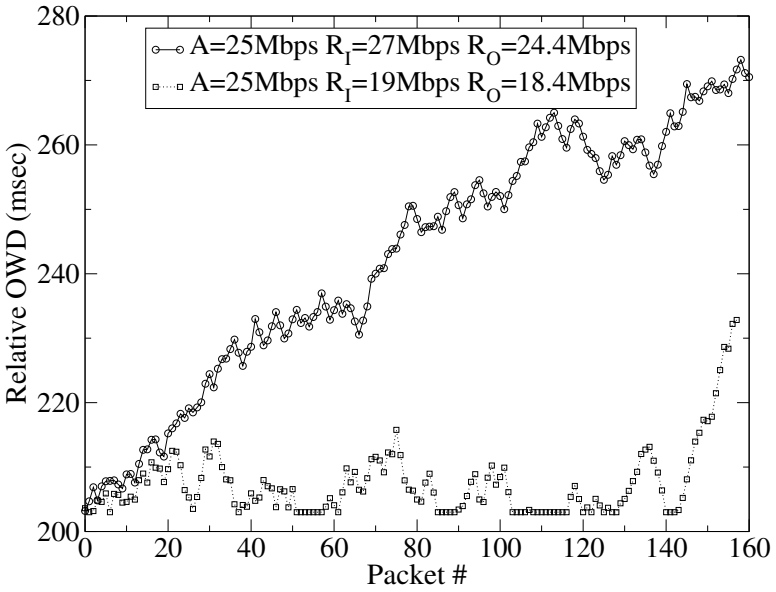


Figure 72: OWDs for two probing streams of 160 packets.

To illustrate, Figure 72 shows the OWDs for two probing streams of 160 packets. In the lower time series, we have that $R_o < R_i$ even though $R_i < A$. The decreased output rate is due to a sudden increase in the OWDs at the very end of the stream (probably due to a cross traffic burst). On the other hand, even a simplistic analysis of the OWDs in that stream would detect that there is no OWD *increasing trend*, inferring correctly that $R_i < A$.

The time series at the top is an example of a clearly increasing OWD trend. In that case, both the OWD trend and the ratio R_o/R_i would correctly infer that $R_i > A$.

7.3.7 Fallacy: Iterative probing converges to a single avail-bw estimate (as opposed to a variation range).

The fact is that iterative probing converges to an avail-bw range rather than to a single value. To see this point, recall that with iterative probing each stream of duration τ and rate $R_i(k)$ can only show if $R_i(k) > A(t, t+\tau)$ or not, where t is the arrival time of the probing stream at the tight link. As we iterate, sending streams with different rates, the avail-bw process $A^\tau(t)$ changes. Consequently, even though a rate $R_i(k)$ may be larger than $A(t_1, t_1+\tau)$, it may be less than $A(t_2, t_2+\tau)$ at some later time t_2. Eventually, after probing at several rates, we can only determine a range (R_L^τ, R_H^τ) in which the avail-bw process $A^\tau(t)$ varies with time. Here, R_L^τ is the lowest probing rate that lead to increasing OWDs, while R_H^τ was the highest probing rate that did not lead to increasing OWDs. The range (R_L^τ, R_H^τ) can be viewed as an estimate of the absolute variation range of the process $A^\tau(t)$ during the measurements.

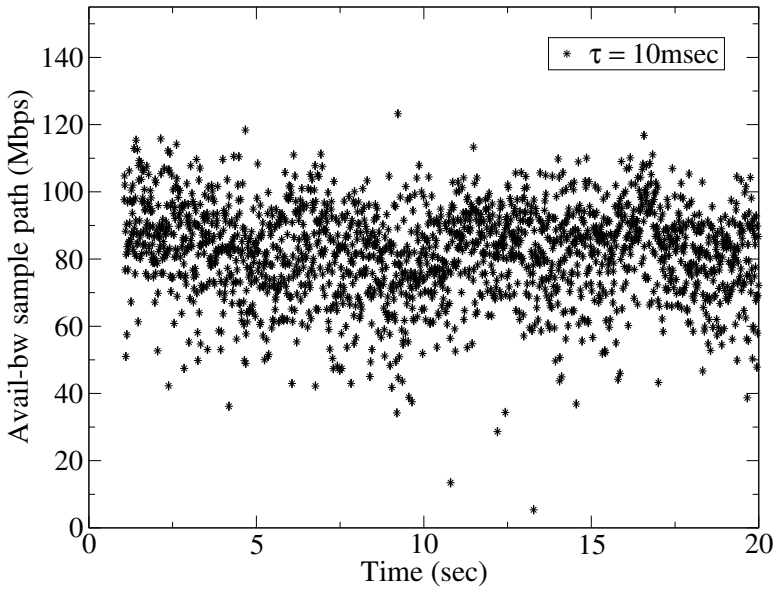


Figure 73: Variation range of an avail-bw sample path.

To illustrate, Figure 73 shows a sample path of avail-bw measurements using the previously mentioned packet trace. A (passive) measurement is obtained in every τ=10ms. Note that the avail-bw in that time scale varies, with significant probability, between 60Mbps and 110Mbps. This range can be estimated with iterative probing, providing us with useful

information regarding, not just the average avail-bw, but the range in which the avail-bw process varies in the time scale that corresponds to the stream duration. Pathload is the only tool that reports such a variation range [33]. Unfortunately, the Pathload range is often misinterpreted as a confidence interval, or as a range-estimate for the average avail-bw.

7.3.8 Pitfall: Ignoring the effects of cross traffic burstiness.

With the fluid model of §7.1, we have that $R_o < R_i$ if and only if $R_i > A$. Due to the cross traffic burstiness, however, a queue can build up at the tight link during the probing stream even if $R_i < A$. This should not be a surprise. We know from basic queuing theory that queues build up even before a server is 100% utilized, depending on the burstiness (variance) and correlation structure of the arriving load.

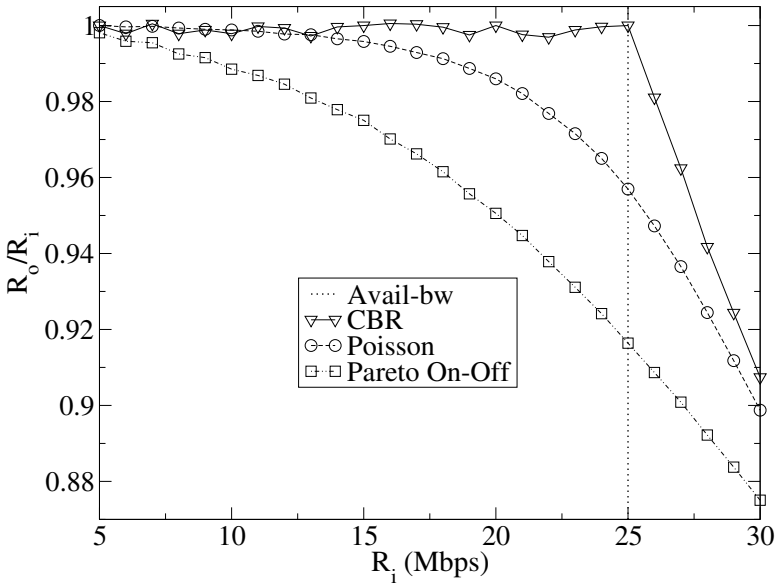


Figure 74: Effect of cross traffic burstiness.

Returning to the previous single-hop simulation setting, we measured the ratio R_o/R_i as a function of R_i for three different cross traffic models: Constant-Bit-Rate (periodic), Poisson, and Pareto ON-OFF[3]. The mean avail-bw is set to 25Mbps. Figure 74 shows the resulting average R_o/R_i ratio from 500 samples. With CBR traffic, we are very close to the fluid model and so the ratio R_o/R_i drops to less than 1.0 only after $R_i > A$. With the two

[3]OFF shape parameter=1.5, ON duration uniformly between 1-10 packets.

other cross traffic models, however, the burstiness of the arriving traffic causes $R_o/R_i < 1$ well before we reach the avail-bw point.

It is clear that the cross traffic burstiness can cause significant underestimation errors in both direct and iterative probing techniques. One may think that the burstiness can be taken into account by using certain thresholds; for instance, to say that $R_i > A$ if R_o/R_i <0.96. These thresholds, however, depend strongly on the measured path and on the cross traffic burstiness, as can be seen comparing the Poisson model with the Pareto ON-OFF model. The fact that traffic burstiness causes underestimation errors in avail-bw estimation has been formally shown recently by Liu *et al.* [54], modeling direct probing in a single-hop scenario.

7.3.9 Pitfall: Ignoring the effects of multiple bottlenecks.

A similar underestimation error can be introduced in the presence of multiple bottlenecks that have approximately equal avail-bw (multiple tight links). The reason is that, in such paths, the more links the probing streams go through, the higher the probability that these streams will interact with cross traffic there, and so the lower their output rate will be.

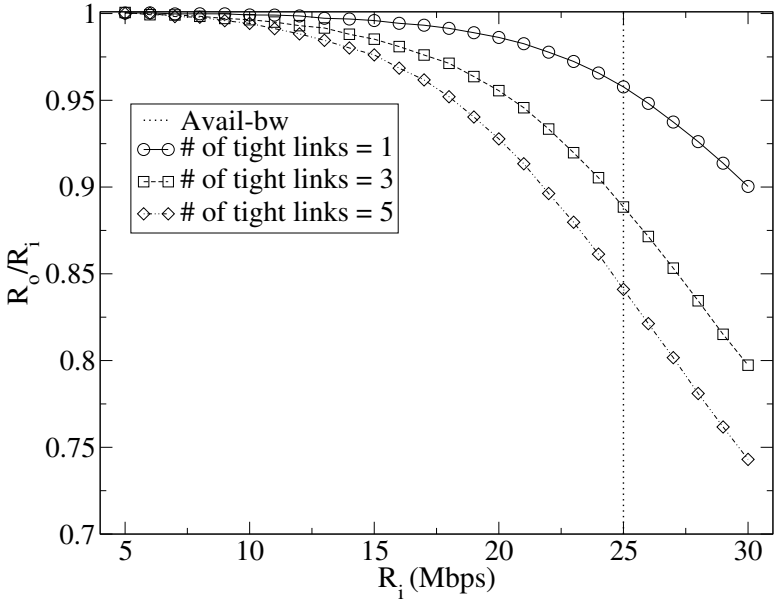


Figure 75: Effect of multiple tight links.

Figure 75 shows simulation results for the average ratio R_o/R_i for 500 samples with

Poisson cross traffic. The cross traffic is one-hop persistent, i.e. it enters the link i and exits at link $i+1$. The main observation is that as the number of tight link increases, the ratio R_o/R_i at the point $R_i = A$ decreases.

The underestimation errors caused by both traffic burstiness and multiple bottlenecks can be viewed as artifacts of the simplistic avail-bw definition in (25). If the avail-bw was defined so that it decreases with the burstiness of the traffic, or with the number of tight links, then the previous underestimation errors could potentially be reduced or avoided. For instance, the *effective bandwidth* metric of [44] captures the maximum load in a queue that does not violate a certain delay or loss rate constraint, considering the traffic burstiness. Such different avail-bw definitions, however, would be less practical, because they would involve the characteristics of the traffic and the load at each link of the path.

More recently, a formal analysis explaining the underestimation errors in the multi-hop path appeared in [55, 52]. Liu et al. [55] showed analytically that underestimation errors in the multi-hop path can result from two factors: cross-traffic burtiness and assumption of single queue path. Furthermore, the underestimation errors in *iterative probing* class of tools can be mitigated by the use of longer packet trains as these tools are only affected by cross traffic burstiness. However, as *direct probing* class of tools are affected by both the factors in multi-hop path, it is not possible to overcome errors in them. Lao et al. [52] presented a more detailed analysis explaining the under estimation problem in *iterative probing* tools.

7.3.10 Pitfall: Evaluating the accuracy of avail-bw estimation through comparisons with bulk TCP throughput.

The related literature makes comparisons between the estimated avail-bw and measured bulk TCP throughput, as a way to verify or evaluate avail-bw estimation techniques. These two metrics however are very different, and they should not be expected to be equal. The throughput of a bulk TCP transfer depends on a number of factors, including socket buffer sizes at the sender and receiver, avail-bw, amount of buffering in the tight link, type of competing cross traffic, round-trip time, loss rate, and several more. Even though the avail-bw is one of these factors, it is certainly not the case that it dominates the others.

We next show some simulation results on the effect of the receiver's advertised window

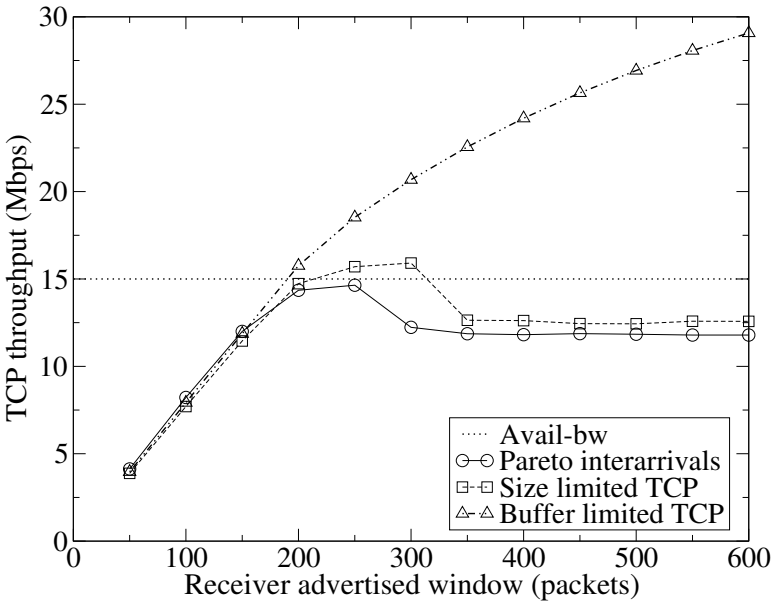


Figure 76: TCP throughput compared to avail-bw.

W_r, as well as on the effect of the cross traffic *congestion responsiveness* on TCP throughput. Figure 76 shows the throughput of a bulk TCP transfer as a function of W_r for three types of cross traffic. The avail-bw is 15Mbps. The cross traffic types are: UDP sources with Pareto interarrivals, a few persistent TCP transfers limited by their advertised windows, and an aggregate of many short TCP transfers. Note that the difference between the avail-bw and the TCP throughput can be positive or negative, and it strongly depends on the congestion responsiveness of the cross traffic, and on W_r. An extended study of the relation between avail-bw and TCP throughput can be found in [37].

7.4 Summary

This chapter reviewed recent work in the area of avail-bw estimation. Additionally, it also identified a number of important misconceptions about avail-bw estimation.

CHAPTER VIII

CONTRIBUTIONS AND FUTURE WORK

8.1 Research Contributions

The e2e argument has shaped the Internet's design for the last 20-30 years. The *best effort delivery* model of the Internet makes no guarantees about the throughout, delay or loss rate of a network flow. In this context, e2e estimation methodologies to infer network state make the design and deployment of advanced services, such as enhanced media services, feasible in the Internet. In particular, the avail-bw, which is a direct indicator of traffic load, has been recognized as a useful metric to characterize the network state for over a decade. However, its e2e estimation has remained an unsolved problem until recently.

In this thesis, we have successfully developed an e2e methodology, called Self-Loading Periodic Streams (SLoPS), for measuring the avail-bw of a network path. SLoPS is based on a simple idea: the one-way delays of a periodic packet stream show an increasing trend when the stream's rate is higher than the avail-bw. As an e2e measurement methodology, SLoPS can have numerous applications such as verification of SLAs, routing in overlay networks and rate adaptation for video streaming.

A major contribution of this thesis is the design and the implementation of algorithms, based on SLoPS, to measure the average avail-bw and avail-bw variation range. The first tool, called *Pathload*, implements an iterative algorithm using SLoPS to estimate the average avail-bw. The second tool, called *Pathvar*, implements two estimation algorithms to estimate the variation range of the avail-bw. We have validated the accuracy of the tools using analysis, simulation, and extensive experimentation. Pathload, in particular, has also been experimentally verified by other researchers and has been shown to be the most accurate tool in comparison with other tools. Pathload has been downloaded by more than 6000 users since 2003. Figure 77 shows the download statistics for Pathload since April 2003.

We have identified that Interrupt Coalescence (IC), a technique implemented in network

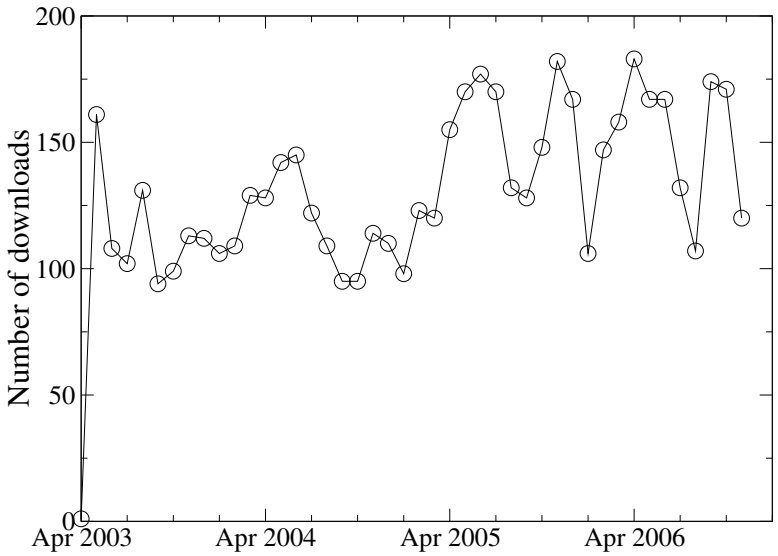


Figure 77: Number of Pathload downloads by unique users since April 2003.

cards to reduce interrupt overhead, is a factor affecting the accuracy of most avail-bw estimation tools. Based on the signature that IC leaves on the dispersion and one-way delays of packet streams, we developed an algorithm to detect IC and to filter its effects from raw measurements.

Finally, we have shown an application of avail-bw estimation to improve the perceived quality of video streaming in the Internet. Our approach is to use overlay path selection to maximize video quality. We show that the avail-bw based path selection scheme yields better perceived video quality than path selection algorithms that rely on jitter or loss-rate measurements. We have also demonstrated that our avail-bw estimation methodologies can be adapted to use application packets, rather than "dummy" probing packets, eliminating the measurement overhead.

8.2 Future Directions

This thesis has developed methodologies and algorithms for avail-bw estimation, which have been deployed both as standalone tools and in an adaptive video streaming application. During the process, we have identified two major issues, discussed in chapter 7. An interesting open question is whether some of the problems we presented, such as the underestimation errors that can be caused due to traffic burstiness or multiple bottlenecks,

can be addressed through improved estimation techniques or through statistical processing of the resulting delay variations.

The second direction emerges the performance of various measurement-driven path selection techniques for video streaming. Our current approach involved measuring the perceived video quality in an offline mode, i.e., after the complete video stream was received; and the path switching decision was based entirely on network path performance. A possible extension is to develop path switching mechanisms that are driven by direct video quality measurements at the receiver instead of network path performance. In addition to being able to measure video quality in realtime, this would require modeling the correlation between network path performance and the perceived video quality. Another open problem is to evaluate the performance of a hybrid approach using both FEC and path selection mechanisms.

Finally, the third direction is to integrate avail-bw estimation techniques with actual applications, and to examine the effectiveness of these techniques given the actual accuracy and latency constraints of real applications. We examined one such application, video streaming, in this thesis. We believe that there are several more applications, such as SLA verification, overlay routing, server selection and network anomaly detection, that can benefit from the use of avail-bw estimation. Using avail-bw estimation for these applications would, however, require novel tools that can measure avail-bw non-intrusively (possibly in a passive manner) and/or are only single-ended. For example, a recent tool called abget [6], which is based on our SLoPS methodology, can form the basis of a server selection scheme. Integrating avail-bw estimation with these applications, however, would also necessitate addressing issues of scalability and/or synchronization among various nodes that use avail-bw information. A recent work by Gao et al. examined the synchronization problem in the context of multihomed networks [21] and proposed randomization techniques combined with avail-bw estimation to avoid oscillations. Similarly, more research in other applications avail-bw estimation might prove valuable in developing novel solutions to improve application performance.

REFERENCES

[1] "Objective Perceptual Video Quality Measurement Techniques for Digital Cable Television in the presence of Full Reference." ITU-T Recommendation J.144 rev. 1, 2003.

[2] "Pathrate and Pathload." http://www.pathrate.org/, Mar. 2007.

[3] Akella, A., Seshan, S., and Shaikh, A., "An Empirical Evaluation of Wide-Area Internet Bottlenecks," in *Proceedings of ACM/USENIX Internet Measurement Conference (IMC)*, 2003.

[4] Allman, M. and Paxson, V., "On Estimating End-to-End Network Path Properties," in *Proceedings of ACM SIGCOMM*, pp. 263–274, Sept. 1999.

[5] Amir, Y., Danilov, C., Goose, S., Hedqvist, D., and Terzis, A., "1-800-OVERLAYS: Using Overlay Networks to Improve VoIP Quality," in *Proceedings of NOSSDAV*, 2005.

[6] Antoniades, D., Athanatos, M., Papagiannakis, A., Markatos, E., and Dovrolis, C., "Available Bandwidth Measurement as Simple as Running Wget," in *Proceedings of PAM*, Apr. 2006.

[7] Apostolopoulos, J., Wong, T., Tan, W., and Wee, S., "On Multiple Description Streaming with Content Delivery Networks," in *Proceedings of INFOCOM*, 2002.

[8] Balk, A., Maggiorini, D., Gerla, M., and Sanadidi, M. Y., "Adaptive MPEG-4 Video Streaming with Bandwidth Estimation," in *Proceedings of QoS-IP*, 2003.

[9] Begen, A., Altunbasak, Y., Ergun, O., and Ammar, M., "Multi-path Selection for Multiple Description Video Streaming over Overlay Networks," *Signal Processing: Image Communication*, vol. 20, pp. 39–60, 2005.

[10] Blake, S., D.Black, M.Carlson, E.Davies, Z.Wang, and W.Weiss, *An Architecture for Differentiated Services*, Dec. 1998. IETF RFC 2475.

[11] Bolch, G., S.Greiner, H.Meer, and K.S.Trivedi, *Queueing Networks and Markov Chains*. John Wiley and Sons, 1999.

[12] Boyce, J. and Gaglianello, R., "Packet Loss Effects on MPEG Video Sent Over the Public Internet," in *Proceedings of Multimedia*, 1998.

[13] Braden, R., D.Clark, and Shenker, S., *Integrated Services in the Internet Architecture: an Overview*, July 1994. RFC 1633.

[14] Carter, R. L. and Crovella, M. E., "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Performance Evaluation*, vol. 27,28, pp. 297–318, 1996.

[15] Cherry, S., "The Battle for Broadband," *IEEE Spectrum*, vol. 42, pp. 24–29, Jan. 2005.

[16] Dovrolis, C., Ramanathan, P., and Moore, D., "What do Packet Dispersion Techniques Measure?," in *Proceedings of IEEE INFOCOM*, pp. 905–914, Apr. 2001.

[17] Dovrolis, C., Ramanathan, P., and Moore, D., "Packet Dispersion Techniques and a Capacity Estimation Methodology," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 963–977, Dec. 2004.

[18] Downey, A., "Using Pathchar to Estimate Internet Link Characteristics," in *Proceedings of ACM SIGCOMM*, pp. 222–223, Sept. 1999.

[19] Endace, "Endace Measurement Systems." http://www.endace.com/, Mar. 2007.

[20] Fenimore, C., "Mastering and Archiving Uncompressed Digital Video Test Materials," in *Proceedings of 142nd SMPTE Technical Conference*, 2000.

[21] Gao, R., Dovrolis, C., and Zegura, E., "Avoiding Oscillations due to Intelligent Route Control Systems," in *Proceedings of IEEE INFOCOM*, 2006.

[22] GRNET, "Greek Research and Technology Network." http://netmon.grnet.gr/traffic/, Mar. 2007.

[23] Guerrero, C. D. and Labrador, M. A., "Experimental and Analytical Evaluation of Available Bandwidth Estimation Tools," in *Proceedings of LCN*, 2006.

[24] Hu, N., Li, L., Mao, Z. M., Steenkiste, P., and Wang, J., "Locating Internet Bottlenecks: Algorithms, Measurements, and Implications," in *Proceedings of ACM SIGCOMM*, Aug. 2004.

[25] Hu, N. and Steenkiste, P., "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 879–894, Aug. 2003.

[26] Hu, N. and Steenkiste, P., "Exploiting Internet Route Sharing for Large Scale Available Bandwidth Estimation," in *Proceedings of ACM IMC*, Oct. 2005.

[27] Intel, "Interrupt Moderation Using Intel Gigabit Ethernet Controllers." http://www.intel.com/design/network/applnots/ap450.pdf, Sept. 2003.

[28] Intel, "Intel Gigabit Ethernet Driver." http://sourceforge.net/projects/e1000, Feb. 2004.

[29] ISC, "Internet Systems Consortium." http://www.isc.org/index.pl?/ops/ds/, Mar. 2007.

[30] Jacobson, V., "Congestion Avoidance and Control," in *Proceedings of ACM SIGCOMM*, Sept. 1988.

[31] Jacobson, V., "Pathchar: A Tool to Infer Characteristics of Internet Paths." ftp://ftp.ee.lbl.gov/pathchar/, Apr. 1997.

[32] Jain, M. and Dovrolis, C., "Pathload: A Measurement Tool for End-to-End Available Bandwidth," in *Proceedings of Passive and Active Measurements (PAM) Workshop*, pp. 14–25, Mar. 2002.

[33] Jain, M. and Dovrolis, C., "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 537–549, Aug. 2003.

[34] Jain, M. and Dovrolis, C., "Ten Fallacies and Pitfalls in End-to-End Available Bandwidth Estimation," in *Proceedings of ACM/USENIX Internet Measurement Conference (IMC)*, Oct. 2004.

[35] Jain, M. and Dovrolis, C., "End-to-end Estimation of Available Bandwidth Variation Range," in *Proceedings of SIGMETRICS*, June 2005.

[36] Jain, M. and Dovrolis, C., "End-to-end Estimation of the Available Bandwidth Variation Range (extended version)," tech. rep., Georgia Tech CERCS, 2005. www.cercs.gatech.edu/tech-reports/.

[37] Jain, M., Prasad, R. S., and Dovrolis, C., "The TCP Bandwidth-Delay Product Revisited: Network Buffering, Cross Traffic, and Socket Buffer Auto-Sizing," Tech. Rep. GIT-CERCS-03-02, Georgia Tech, Feb. 2003. Available at http://www.cercs.gatech.edu/tech-reports/.

[38] Jeffay, K., Stone, D. L., Talley, T., and Smith, F. D., "Adaptive, Best-Effort Delivery of Digital Audio and Video Across Packet-Switched Networks," in *Network and Operating System Support for Digital Audio and Video*, 1993.

[39] Jiang, W. and Schulzrinne, H., "Comparison and Optimization of Packet Loss Repair Methods on VOIP Perceived Quality under Bursty Loss," in *Proceedings of NOSSDAV*, 2002.

[40] Jin, G. and Tierney, B. L., "System Capability Effects on Algorithms for Network Bandwidth Measurement," in *Proceedings of ACM Internet Measurement Conference*, Oct. 2003.

[41] Jin, G., Yang, G., Crowley, B., and Agarwal, D., "Network Characterization Service (NCS)," in *Proceedings of 10th IEEE Symposium on High Performance Distributed Computing*, Aug. 2001.

[42] Kanakia, H., Mishra, P., and Reibman, A., "An Adaptive Congestion Control Scheme for Real Time Packet Video Transport," *IEEE Transactions of Networking*, vol. 3, no. 6, 1995.

[43] Katabi, D., Handley, M., and Rohrs, C., "Congestion Control for High Bandwidth-Delay Product Networks," in *Proceedings of ACM SIGCOMM*, Aug. 2002.

[44] Kelly, F., *Stochastic Networks: Theory and Applications*, ch. Notes on effective bandwidths, pp. 141–168. Oxford University Press, 1996.

[45] Keshav, S., "A Control-Theoretic Approach to Flow Control," in *Proceedings of ACM SIGCOMM*, pp. 3–15, Sept. 1991.

[46] Kilpi, J. and Norros, I., "Testing the Gaussian Approximation of Aggregate Traffic," in *Proceedings of ACM/USENIX Internet Measurement Workshop (IMW)*, Nov. 2002.

[47] Kontothanassis, L., Sitaraman, R., Wein, J., Hong, D., Kleinberg, R., Mancuso, B., Shaw, D., and Stodolsky, D., "A Transport Layer for Live Streaming in a Content Delivery Network," *Proceedings of IEEE*, vol. 92, 2004.

[48] Lai, K. and Baker, M., "Measuring Bandwidth," in *Proceedings of IEEE INFOCOM*, pp. 235–245, Apr. 1999.

[49] Lai, K. and Baker, M., "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay," in *Proceedings of ACM SIGCOMM*, pp. 283–294, Sept. 2000.

[50] Lakshman, T. V. and Madhow, U., "The Performance of TCP/IP for Networks with High Bandwidth- delay Products and Random Losses," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 336–350, June 1997.

[51] Lam, W. M. and Reibman, A. R., "An Error Concealment Algorithm for Images Subject to Channel Errors," *IEEE Transactions on Image Processing*, vol. 4, pp. 533–542, May 1995.

[52] Lao, L., Dovrolis, C., and Sanadidi, M., "The Probe Gap Model can Underestimate the Available Bandwidth of Multi-hop Paths," *ACM CCR*, vol. 36, Oct. 2006.

[53] Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V., "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 1–15, Feb. 1994.

[54] Liu, X., Ravindran, K., Liu, B., and Loguinov, D., "Single-Hop Probing Asymptotics in Available Bandwidth Estimation: A Sample-Path Analysis," in *Proceedings of ACM Internet Measurement Conference*, 2004.

[55] Liu, X., Ravindran, K., and Loguinov, D., "Multi-Hop Probing Asymptotics in Available Bandwidth Estimation: Stochastic Analysis," in *Proceedings of ACM Internet Measurement Conference*, Oct. 2005.

[56] Lu, X., Tao, S., Zarki, M. E., and Guerin, R., "Quality-Based Adaptive Video Over the Internet," in *Proceedings of CNDS*, 2003.

[57] Mah, B. A., "pchar: a Tool for Measuring Internet Path Characteristics." http://www.employees.org/∼bmah/Software/pchar/, Feb. 1999.

[58] Mathis, M. and Allman, M., *A Framework for Defining Empirical Bulk Transfer Capacity Metrics*, July 2001. RFC 3148.

[59] Melander, B., Bjorkman, M., and Gunningberg, P., "A New End-to-End Probing and Analysis Method for Estimating Bandwidth Bottlenecks," in *IEEE Global Internet Symposium*, 2000.

[60] Melander, B., Bjorkman, M., and Gunningberg, P., "Regression-Based Available Bandwidth Measurements," in *International Symposium on Performance Evaluation of Computer and Telecommunications Systems*, 2002.

[61] Mogul, J. C. and Ramakrishnan, K. K., "Eliminating receive livelock in an interrupt-driven kernel," *ACM Transactions on Computer Systems*, vol. 15, pp. 217–252, Aug. 1997.

[62] Nichols, K., Blake, S., Baker, F., and Black, D. L., *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, Dec. 1998. IETF RFC 2474.

[63] NLANR MOAT, "Passive Measurement and Analysis." http://pma.nlanr.net/pma/, Dec. 2006.

[64] NSv2, "Network Simulator." http://www.isi.edu/nsnam/ns/, Dec. 2006.

[65] Oetiker, T., "MRTG: Multi Router Traffic Grapher." http://oss.oetiker.ch/mrtg/, Mar. 2007.

[66] Park, K. and W. Willinger (editors), *Self-Similar Network Traffic and Performance Evaluation*. John Willey, 2000.

[67] Pasztor, A., *Accurate Active Measurement in the Internet and its Applications*. PhD thesis, The University of Melbourne, Feb. 2003.

[68] Pasztor, A. and Veitch, D., "The Packet Size Dependence of Packet Pair Like Methods," in *IEEE/IFIP International Workshop on Quality of Service (IWQoS)*, 2002.

[69] Paxson, V., "On Calibrating Measurements of Packet Transit Times," in *Proceedings of ACM SIGMETRICS*, pp. 11–21, June 1998.

[70] Paxson, V., "End-to-End Internet Packet Dynamics," *IEEE/ACM Transaction on Networking*, vol. 7, pp. 277–292, June 1999.

[71] Pinson, M. and Wolf, S., "NTIA HB 06434: In-Service Video Quality Metric Users Manual." http://www.its.bldrdoc.gov/pub/ntia-rpt/06-434/, Apr. 2005.

[72] Prasad, R. S., Jain, M., and Dovrolis, C., "Effects of Interrupt Coalescence on Network Measurements," in *Proceedings Passive and Active Measurements (PAM) workshop*, Apr. 2004.

[73] Prasad, R. S., Murray, M., Dovrolis, C., and Claffy, K., "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," *IEEE Network*, Nov. 2003.

[74] Ramakrishnan, K., Floyd, S., and Black, D., *The Addition of Explicit Congestion Notification (ECN) to IP*, Sept. 2001. RFC 3168.

[75] Reed, I. and Solomon, G., "Polynomial Codes over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, 1960.

[76] Rejaie, R., Handley, M., and Estrin, D., "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proceedings of IEEE INFOCOM*, 1999.

[77] Ribeiro, V., Coates, M., Riedi, R., Sarvotham, S., Hendricks, B., and Baraniuk, R., "Multifractal Cross-Traffic Estimation," in *Proceedings ITC Specialist Seminar on IP Traffic Measurement, Modeling, and Management*, Sept. 2000.

[78] Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., and Cottrell, L., "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," in *Proceedings of Passive and Active Measurements (PAM) workshop*, Apr. 2003.

[79] Salama, P., Shroff, N., Coyle, E., and Delp, E., "Error Concealment Techniques for Encoded Video Streams," in *Proceedings IEEE International Conference on Image Processing*, pp. 9–12, 1995.

[80] Shriram, A., Murray, M., Hyun, Y., Brownlee, N., Broido, A., Fomenkov, M., and Claffy, K. C., "Comparison of Public End-to-End Bandwidth Estimation Tools on High Speed Links," in *Proceedings of PAM*, 2005.

[81] Strauss, J., Katabi, D., and Kaashoek, F., "A Measurement Study of Available Bandwidth Estimation Tools," in *Proceedings of ACM/USENIX Internet Measurement Conference (IMC)*, 2003.

[82] Syskonnect, "SK-NET GE Gigabit Ethernet Server Adapter." http://www.syskonnect.com/syskonnect/technology/SK-NET_GE.PDF, June 2003.

[83] Syskonnect, "SysKonnect Gigabit Ethernet Driver." http://www.syskonnect.com/syskonnect/support/driver/ge.htm, June 2003.

[84] Tan, W. and Zakhor, A., "Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol," *IEEE Transactions on Multimedia*, vol. 1, no. 2, 1999.

[85] Tao, S., Apostopoulos, J., and Guerin, R., "Real-Time Monitoring of Video Quality in IP Networks," in *Proceedings of NOSSDAV*, 2005.

[86] Tao, S. and Guerin, R., "Application-specific Path Switching: A Case Study for Streaming Video," in *Proceedings of ACM International Conference on Multimedia*, 2004.

[87] Tian, X., Wu, J., and Ji, C., "A Unified Framework for Understanding Network Traffic Using Independent Wavelet Models," in *Proceedings of IEEE INFOCOM*, June 2002.

[88] VLC Media Player. http://www.videolan.org/vlc, Dec. 2006.

[89] White, P. P., "RSVP and Integrated Services in the Internet: A Tutorial," *IEEE Communications Magazine*, pp. 100–106, May 1997.

[90] Wolf, S., "VQM Software." http://www.its.bldrdoc.gov/n3/video/vqmsoftware.htm, Dec. 2006.

[91] Wroclawski, J., *The Use of RSVP with IETF Integrated Services*, Sept. 1997. RFC 2210.

[92] Zec, M., Mikuc, M., and Zagar, M., "Estimating the Impact of Interrupt Coalescing Delays on Steady State TCP Throughput," in *Proceedings of 10-th SoftCOM*, 2002.

[93] Zhou, J., Shao, H., Shen, C., and Sun, M., "Multi-path Transport of FGS Video," in *Proceedings of Packet Video Workshop*, 2003.

[94] Zhu, Y., Dovrolis, C., and Ammar, M., "Dynamic Overlay Routing Based on Available Bandwidth Estimation: A Simulation Study," *Computer Networks Journal*, 2006.

[95] Zukerman, M., Neame, T. D., and Addie, R. G., "Internet Traffic Modeling and Future Technology Implications," in *Proceedings of IEEE INFOCOM*, 2003.

VITA

Manish Jain received the B.Engg. degree in Computer Engineering from SGSITS, Indore, India, in 1999, and the M.S. degree in Computer Science from the University of Delaware, Newark, DE, in 2002. He joined Georgia Tech in Fall 2002 and finished his doctorate under the able guidance of Dr. Constantine Dovrolis on the topic of *End-to-end Available Bandwidth Estimation and its Applications*. He has worked as a summer intern at HP Labs, Palo Alto in the summer of 2005. His current research interests include bandwidth estimation methodologies and applications, advanced media services in IP networks, and network problem diagnostics.